



**UNIVERSIDADE ESTADUAL DE CAMPINAS  
FACULDADE DE ENGENHARIA CIVIL,  
ARQUITETURA E URBANISMO**

**Aplicação de Redes Neurais Artificiais em Simulação  
Numérica do Acoplamento Poço-Reservatório**

**Thiago Dias dos Santos**

Campinas  
2012

**UNIVERSIDADE ESTADUAL DE CAMPINAS  
FACULDADE DE ENGENHARIA CIVIL, ARQUITETURA E URBANISMO**

**Aplicação de Redes Neurais Artificiais em Simulação Numérica do  
Acoplamento Poço-Reservatório**

**Thiago Dias dos Santos**

Dissertação apresentada à Comissão de Pós-graduação da Faculdade de Engenharia Civil, Arquitetura e Urbanismo da Universidade Estadual de Campinas, como parte dos requisitos para obtenção do título de Mestre em Engenharia Civil, na área de concentração de Estruturas.

**Orientador: Prof. Dr. Philippe Remy Bernard Devloo**

Campinas  
2012

FICHA CATALOGRÁFICA ELABORADA PELA  
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Sa59a	<p>Santos, Thiago Dias dos Aplicação de redes neurais artificiais em simulação numérica do acoplamento poço-reservatório / Thiago Dias dos Santos. --Campinas, SP: [s.n.], 2012.</p> <p>Orientador: Philippe Remy Bernard Devloo. Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Civil, Arquitetura e Urbanismo.</p> <p>1. Redes neurais artificiais. 2. Engenharia de petróleo. 3. Metodos de simulação. 4. Simulação por computador. 5. Analise numérica. I. Devloo, Philippe Remy Bernard. II. Universidade Estadual de Campinas. Faculdade de Engenharia Civil, Arquitetura e Urbanismo. III. Título.</p>
-------	---

Título em Inglês: Artificial neural networks applied to the numerical simulation of well-reservoir coupling

Palavras-chave em Inglês: Artificial neural network, Petroleum engineering, Simulation methods, Computer simulation, Numerical analysis

Área de concentração: Estruturas

Titulação: Mestre em Engenharia Civil

Banca examinadora: Mario Conrado Cavichia, Romis Ribeiro de Faissol Attux

Data da defesa: 31-01-2012

Programa de Pós Graduação: Engenharia Civil

**UNIVERSIDADE ESTADUAL DE CAMPINAS  
FACULDADE DE ENGENHARIA CIVIL, ARQUITETURA E URBANISMO**

**Aplicação de Redes Neurais Artificiais em Simulação Numérica do  
Acoplamento Poço-Reservatório**

**Thiago Dias dos Santos**

**Dissertação de Mestrado aprovada pela Banca Examinadora, constituída por:**

---

Prof. Dr. Philippe Remy Bernard Devloo  
**Presidente e Orientador / FEC - UNICAMP**

---

Prof. Dr. Mario Conrado Cavichia  
**FEC - UNICAMP**

---

Prof. Dr. Romis Ribeiro de Faissol Attux  
**FEEC - UNICAMP**

Campinas, 31 de janeiro de 2012



À minha família, exemplo de amor,

humildade e, principalmente, fé:

Gervásio Bento dos Santos,

Ph.D. na arte de doar a vida pela família.

Benedita Lopes Dias dos Santos,

Ph.D. na arte da obediência e da humildade.

Jacqueline Marques do Santos,

doutoranda na arte de conquistar pessoas.

Kevin Caio Marques dos Santos,

mestrando na arte de aprender e de ser companheiro.



# Agradecimentos

Agradeço a Deus, "eterna verdade, verdadeira caridade e querida eternidade!" (Sto Agostinho).

Agradeço especialmente aos enormes companheiros de batalha, os quais me ensinaram não só o conhecimento técnico, mas também peculiaridades que somente a vida pode nos dar. São eles:

- Orientador Philippe Remy Bernard Devloo, *engemático* de ideias e de soluções  $n$ -dimensionais complexas, ou perplexas, dependendo do ponto de vista operacional;
- Professora Silvana Bastos, UFPE, pelas sugestões e auxílio na pesquisa em redes neurais artificiais;
- Professor Francisco Antonio Menezes, grande amigo que tem me acompanhado na carreira acadêmica desde as primeiras iniciações científicas e grande exemplo profissional;
- Amigos de estudo e trabalho do LabMeC: Jorge Calle, professora Sônia Maria Gomes, Tiago Forti, Edimar Rylo, Gustavo Longhin, Alaor Rosa, Maurício Souza, Agnaldo Farias, Denise Siqueira, João Gonçalves, Caju, Diogo Cecílio, Nathan Shauer, Mariane Montibeller Silva, Ana Clara Galindo Rosa, Tiago Almeida, José Antonio Silverio, Tito Rezende;
- Pessoal do LAPLA, pela amizade, solidarização e *cafezinhos* compartilhados. Especialmente: Isadora Salviano e Dani Lins.;
- Amigos de todas as bandas e *quantas*. Especialmente Giovanna Trevizan, pela amizade e pela mútua solidarização frente às dificuldades acadêmicas; pessoal do TLC, por toda força e fé ao longo dos últimos meses; ex-companheiros da FEC, pelo incentivo ao desenvolvimento profissional e acadêmico;
- FEC - UNICAMP;
- CENPES - PETROBRAS, ANP;
- SimWorx.



***De um calculista que quase ficou doidão***

*Quando era criancinha acreditava em  
Papai Noel, Sacizinho e Bicho Papão.  
Já na mocidade aprendi na faculdade,  
com dificuldade, a tal Singularidade.*

*Depois, como balela, vi com uma tabela  
Crescimento de Ferros Dobrados a Mão  
Também lá, os mestres me ensinaram até  
Tombamento de Muro em Torno do seu Pé.*

*E pra me graduar tive que tolerar,  
em um sermão, que Viga Trabalha a Torção  
**40 anos depois...***

*Hoje com um bom lampião e sagacidade,  
caço a singularidade e a viga a torção.*

*Porém uma não resiste à realidade  
e a outra não tolera deformação.*

*Da balela dos ferros que crescem na mão,  
tem quem gosta mas não servem pra construção.*

*Dos muros tombados, fora os que escorregaram  
bem apoiados sobre estacas em seu pé,  
sobram os que o seu bom sub-solo esmagaram,  
pois tão duro chão só mesmo com muita fé.*

*Finalizando, saber com seriedade  
só se adquire com muita maturidade.*



# Resumo

No presente trabalho, desenvolveu-se uma biblioteca para geração de redes neurais artificiais (*NeuralLib*) e aplicou-se a mesma para aproximação do acoplamento de escoamento em poços horizontais com reservatório. A biblioteca *NeuralLib* foi desenvolvida em linguagem C++. A arquitetura de rede gerada e utilizada foi a *Multilayer Perceptron (MLP)* com uma única camada oculta. Optou-se em gerar 3 arquiteturas com diferentes números de neurônios ocultos com objetivo de analisar o comportamento das *MLPs*. O algoritmo de treinamento adotado foi o de retropropagação ou *backpropagation*.

A rede neural foi utilizada para mapear o fluxo do reservatório tridimensional para o poço horizontal. O escoamento no poço é simulado utilizando leis constitutivas turbulentas e laminares. Foi elaborada uma técnica para gerar os conjuntos de padrões para o processo de treinamento das *MLPs*, utilizando para tal as curvas de fluxo do reservatório para o poço provenientes de um modelo tridimensional. As *MLPs* treinadas foram utilizadas na resolução de um modelo unidimensional fornecendo valores de um parâmetro de fluxo do reservatório. Nesse processo, o modelo unidimensional produziu curvas de fluxo no poço semelhantes aos gerados pelo modelo tridimensional. Os resultados são avaliados com relação ao processo de treinamento das *MLPs* e com relação às curvas de fluxo e vazão total de produção dos poços.

**Palavras chave:** Redes Neurais Artificiais, Engenharia de Petróleo, Métdos de Simulação, Simulação por Computador, Análise Numérica.



# Abstract

*In this work, an object-oriented library was developed which implements neural networks (NeuralLib). The library was used to model the coupling of the fluid flow in a three-dimensional reservoir with a one-dimensional well model. The architecture of the neural network is the Multilayer Perceptron (MLP) with a single hidden layer. Three different architectures with varying number of hidden neurons were tested to evaluate the behaviour of the MLP. The backpropagation algorithm was used to train the network.*

*The neural network was applied to estimate the mass flux from a three dimensional reservoir to a horizontal well. The fluid flow in the horizontal well uses laminar and turbulent constitutive models. A technique was developed to generate a set of patterns which were used to train the MLP's. The MLP's output data is a function which represents the mass flux from the reservoir to the one dimensional well. Using the mass flux function, the pressure function in the horizontal well and well flux were very close to the pressure and flux computed using the three dimensional model. The effectiveness of the neural network was evaluated by comparing cases which were not included in the original training set.*

**Keywords:** Artificial Neural Networks, Petroleum Engineering, Simulation Methods, Computer Simulation, Numerical Analysis.



# Sumário

<b>Sumário</b>	<b>xv</b>
<b>Lista de Figuras</b>	<b>xvii</b>
<b>Lista de Tabelas</b>	<b>xix</b>
<b>Lista de Símbolos</b>	<b>xxi</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Apresentação: Redes Neurais Artificiais . . . . .	1
1.2 Motivação . . . . .	2
1.3 Objetivos . . . . .	3
1.4 Organização do Trabalho . . . . .	3
<b>2 Revisão Bibliográfica</b>	<b>5</b>
2.1 Redes Neurais Artificiais . . . . .	5
2.1.1 Definição . . . . .	5
2.1.2 Neurônio Artificial . . . . .	6
2.1.3 Arquiteturas de Redes . . . . .	12
2.1.4 Processos de Aprendizagem . . . . .	15
2.2 Acoplamento Poço-Reservatório . . . . .	20
2.2.1 Modelo Tridimensional do Acoplamento Poço-Reservatório . . . . .	21
2.2.2 Escoamento no Reservatório . . . . .	23
2.2.3 Escoamento no Poço Horizontal . . . . .	24
2.2.4 Curvas de Fluxo do Modelo Tridimensional . . . . .	25
<b>3 Metodologia</b>	<b>29</b>
3.1 Descrição Geral . . . . .	29
3.2 Ferramentas Computacionais Utilizadas . . . . .	30

3.3	Acoplamento Poço-Reservatório: Unidimensional . . . . .	31
3.4	Análise da Resistividade $K(x)$ . . . . .	33
3.4.1	Adimensionalização da Resistividade $K(x)$ . . . . .	34
3.4.2	Representação de $K(x)$ . . . . .	36
3.5	Aprendizagem da Rede Neural Artificial . . . . .	38
3.5.1	Conjunto de Treinamento . . . . .	38
3.5.2	Arquitetura de Rede Neural Utilizada . . . . .	41
3.5.3	Processo de Aprendizagem . . . . .	42
3.6	<i>MLP</i> e o Modelo Unidimensional . . . . .	43
3.6.1	Resolução do Modelo Unidimensional . . . . .	43
3.6.2	Análise dos Resultados: Medida de Erros . . . . .	44
<b>4</b>	<b>Resultados</b>	<b>47</b>
4.1	<i>NeuralLib</i> . . . . .	47
4.2	Aprendizagem da Rede Neural . . . . .	52
4.2.1	Arquitetura 1: <i>MLP-5</i> . . . . .	53
4.2.2	Arquitetura 2: <i>MLP-10</i> . . . . .	55
4.2.3	Arquitetura 3: <i>MLP-15</i> . . . . .	57
4.2.4	Comparação . . . . .	59
4.3	Redes Neurais e o Modelo Unidimensional . . . . .	61
4.3.1	<i>MLP-5</i> e Modelo Unidimensional . . . . .	61
4.3.2	<i>MLP-10</i> e Modelo Unidimensional . . . . .	63
4.3.3	<i>MLP-15</i> e Modelo Unidimensional . . . . .	65
4.3.4	Comparação . . . . .	66
<b>5</b>	<b>Conclusão</b>	<b>77</b>
<b>Referências Bibliográficas</b>		<b>79</b>
<b>A</b>	<b><i>NeuralLib</i></b>	<b>83</b>
A.1	Descrição das Classes . . . . .	83
A.2	Caso de Uso . . . . .	86
A.3	Arquivo Treinamento . . . . .	90

# Listas de Figuras

2.1	Modelo simplificado do neurônio de <i>McCulloch e Pitts</i> . . . . .	7
2.2	Função de limiar. . . . .	9
2.3	Função linear por partes. . . . .	10
2.4	Função logística. . . . .	10
2.5	Função tangente hiperbólica. . . . .	11
2.6	Exemplo de função de base radial: função <i>Gaussiana</i> . . . . .	11
2.7	Rede alimentada adiante com uma camada de entrada e uma de saída. . . . .	12
2.8	Rede alimentada adiante com uma camada oculta de neurônios. . . . .	13
2.9	Rede recorrente autoalimentada. . . . .	14
2.10	Rede recorrente com entrada e saída. . . . .	14
2.11	Poço horizontal e reservatório elíptico: elementos tridimensionais curvos. . . . .	22
2.12	<i>Wireframe</i> da malha do poço horizontal e reservatório elíptico. . . . .	22
2.13	Curva de pressão no poço horizontal. . . . .	26
2.14	Curva de vazão no poço horizontal. . . . .	26
2.15	Curva de distribuição de vazão no poço horizontal. . . . .	27
2.16	Valor do número de <i>Reynolds</i> no poço horizontal. . . . .	27
3.1	Arranjo das redes neurais e modelo 1D do acoplamento poço-resevatório . . . . .	30
3.2	Modelo simplificado unidimensional do acoplamento poço-reservatório. . . . .	31
3.3	Exemplo da curva $K(x)$ . . . . .	33
4.1	Função <i>Peaks</i> . . . . .	48
4.2	Comparação entre os gráfico original e o gerado pela <i>MLP</i> . . . . .	48
4.3	Função descontínua. . . . .	49
4.4	Comparação entre os gráfico original e o gerado pela <i>MLP</i> . . . . .	50
4.5	Função <i>Rastrigin</i> . . . . .	51
4.6	Comparação entre os gráfico original e o gerado pela <i>MLP</i> . . . . .	51
4.7	Evolução da energia média do erro do subconjunto $A_T$ ao longo das épocas. . . . .	53

4.8	Evolução da energia média do erro do subconjunto $A_v$ ao longo das épocas. . . . .	53
4.9	Evolução da energia média do erro do subconjunto $A_T$ ao longo das épocas. . . . .	55
4.10	Evolução da energia média do erro do subconjunto $A_v$ ao longo das épocas. . . . .	55
4.11	Evolução da energia média do erro do subconjunto $A_T$ ao longo das épocas. . . . .	57
4.12	Evolução da energia média do erro do subconjunto $A_v$ ao longo das épocas. . . . .	57
4.13	Comparação das energias totais dos erros do conjunto $A$ para as três <i>MLPs</i> . . . . .	59
4.14	Comparação norma $L^2$ do conjunto $A$ para as três <i>MLPs</i> - curva $Q_{hw}(x)$ . . . . .	67
4.15	Comparação norma $L^2$ do conjunto $A$ para as três <i>MLPs</i> - curva $p(x)$ . . . . .	67
4.16	Comparação norma $L^2$ do conjunto $A$ para as três <i>MLPs</i> - curva $dQ_{hw}(x)/dx$ . . . .	68
4.17	Comparação norma $L^2$ do conjunto $A$ para as três <i>MLPs</i> - curva $dp(x)/dx$ . . . . .	68
4.18	Comparação norma $L^1$ dos erros do conjunto $A$ para as três <i>MLPs</i> - $Q_{hw}$ . . . . .	69
4.19	Comparação modelo 3D e <i>MLP+modelo 1D</i> , caso 100 - curva $Q_{hw}(x)$ . . . . .	72
4.20	Comparação modelo 3D e <i>MLP+modelo 1D</i> , caso 100 - curva $p(x)$ . . . . .	72
4.21	Comparação modelo 3D e <i>MLP+modelo 1D</i> , caso 100 - curva $dQ_{hw}(x)/dx$ . . . . .	73
4.22	Comparação modelo 3D e <i>MLP+modelo 1D</i> , caso 100 - curva $dp(x)/dx(x)$ . . . . .	73
4.23	Comparação modelo 3D e <i>MLP+modelo 1D</i> , caso 52 - curva $Q_{hw}(x)$ . . . . .	74
4.24	Comparação modelo 3D e <i>MLP+modelo 1D</i> , caso 52 - curva $p(x)$ . . . . .	74
4.25	Comparação modelo 3D e <i>MLP+modelo 1D</i> , caso 52 - curva $dQ_{hw}(x)/dx$ . . . . .	75
4.26	Comparação modelo 3D e <i>MLP+modelo 1D</i> , caso 52 - curva $dp(x)/dx(x)$ . . . . .	75

# Listas de Tabelas

4.1	Energia média do erro para os subconjuntos $A_T$ , $A_v$ e $A_{te}$ . . . . .	54
4.2	Energia média do erro para os subconjuntos $A_T$ , $A_v$ e $A_{te}$ . . . . .	56
4.3	Energia média do erro para os subconjuntos $A_T$ , $A_v$ e $A_{te}$ . . . . .	58
4.4	Energia média do erro para o subconjunto $A_T$ . . . . .	60
4.5	Energia média do erro para o subconjunto $A_v$ . . . . .	60
4.6	Energia média do erro para o subconjunto $A_{te}$ . . . . .	60
4.7	Norma $L^2$ média do erro e desvio padrão - curva $Q_{hw}(x)$ . . . . .	61
4.8	Norma $L^2$ média do erro e desvio padrão - curva $p(x)$ . . . . .	61
4.9	Norma $L^2$ média do erro e desvio padrão - curva $dQ_{hw}(x)/dx$ . . . . .	62
4.10	Norma $L^2$ média do erro e desvio padrão - curva $dp(x)/dx$ . . . . .	62
4.11	Norma $L^1$ (alterada) média do erro e desvio padrão - $Q_{hw}$ . . . . .	62
4.12	Norma $L^2$ média do erro e desvio padrão - curva $Q_{hw}(x)$ . . . . .	63
4.13	Norma $L^2$ média do erro e desvio padrão - curva $p(x)$ . . . . .	63
4.14	Norma $L^2$ média do erro e desvio padrão - curva $dQ_{hw}(x)/dx$ . . . . .	63
4.15	Norma $L^2$ média do erro e desvio padrão - curva $dp(x)/dx$ . . . . .	63
4.16	Norma $L^1$ (alterada) média do erro e desvio padrão - $Q_{hw}$ . . . . .	64
4.17	Norma $L^2$ média do erro e desvio padrão - curva $Q_{hw}(x)$ . . . . .	65
4.18	Norma $L^2$ média do erro e desvio padrão - curva $p(x)$ . . . . .	65
4.19	Norma $L^2$ média do erro e desvio padrão - curva $dQ_{hw}(x)/dx$ . . . . .	65
4.20	Norma $L^2$ média do erro e desvio padrão - curva $dp(x)/dx$ . . . . .	65
4.21	Norma $L^1$ (alterada) média do erro e desvio padrão - $Q_{hw}$ . . . . .	66
4.22	Comparativo entre os erros das curvas e da vazão total, subconjunto $A_T$ . . . . .	70
4.23	Comparativo entre os erros das curvas e da vazão total, subconjunto $A_v$ . . . . .	70
4.24	Comparativo entre os erros das curvas e da vazão total, subconjunto $A_{te}$ . . . . .	71
A.1	Trecho de um arquivo de treinamento de uma rede neural. . . . .	90



# Listas de Símbolos

$MLP$	(def. pág. 2)	<i>Multilayer Perceptron</i>
$\varphi(\cdot)$	(def. pág. 8)	função de ativação de um neurônio
$u_k$	(def. pág. 8)	saída do combinador linear do neurônio $k$
$y_k$	(def. pág. 8)	saída do neurônio $k$
$w_{kj}$	(def. pág. 8)	peso sináptico do neurônio $k$ para o sinal de entrada $j$
$x_j$	(def. pág. 8)	sinal de entrada $j$
$b_k$	(def. pág. 8)	<i>bias</i> do neurônio $k$
$v_k$	(def. pág. 8)	campo local induzido do neurônio $k$
$r$	(def. pág. 11)	raio de abertura da função de base radial
$\vec{y}$	(def. pág. 16)	vetor sinal de saída da rede
$\vec{d}$	(def. pág. 16)	vetor saída esperada da rede
$d_i$	(def. pág. 16)	valor de saída esperado do neurônio $i$
$\vec{e}$	(def. pág. 16)	vetor sinal de erro da rede
$e_i$	(def. pág. 16)	sinal de erro do neurônio $i$
$\xi$	(def. pág. 16)	valor instantâneo da energia do erro da rede neural
$\xi_{med}$	(def. pág. 17)	valor médio da energia do erro da rede neural
$T$	(def. pág. 17)	dimensão da amostra de treinamento
$\triangle w_{ji}$	(def. pág. 18)	fator de correção do peso sináptico $i$ do neurônio $j$
$\eta$	(def. pág. 18)	taxa de aprendizagem da rede neural
$\delta_j$	(def. pág. 18)	gradiente local para o neurônio $j$
$\varphi'_j(\cdot)$	(def. pág. 18)	derivada da função de ativação do neurônio $j$
$\alpha$	(def. pág. 20)	constante de momento da regra delta generalizada
$Q$	(def. pág. 23)	fluxo de fluido
$\mu$	(def. pág. 23)	viscosidade do óleo
$\overleftrightarrow{K}$	(def. pág. 23)	tensor de permeabilidade do meio poroso

$p$	(def. pág. 23)	pressão do fluido
$\vec{n}$	(def. pág. 23)	vetor normal a uma face
$\sigma_0$	(def. pág. 23)	fluxo normal conhecido numa face (c.c. <i>Neumann</i> )
$p_0$	(def. pág. 23)	pressão conhecida numa face ou região (c.c. <i>Dirichlet</i> )
$x$	(def. pág. 24)	coordenada do eixo longitudinal do poço
$f$	(def. pág. 24)	fator de fricção
$V$	(def. pág. 24)	velocidade média do fluxo na seção circular do poço
$\rho$	(def. pág. 24)	massa específica do fluido
$D$	(def. pág. 24)	diâmetro do poço
$Q_{hw}$	(def. pág. 24)	vazão de fluido no poço horizontal
$q_l$	(def. pág. 24)	fluxo por unidade de comprimento
$Q_{heel}$	(def. pág. 25)	vazão de fluido no poço horizontal (no <i>heel</i> )
$p(x)$	(def. pág. 25)	pressão ao longo do poço
$Q_{hw}(x)$	(def. pág. 26)	vazão ao longo do poço
$\frac{dQ_{hw}(x)}{dx}$	(def. pág. 27)	distribuição de vazão ao longo do poço
$Rey(x)$	(def. pág. 27)	número de <i>Reynolds</i> ao longo do poço
$\frac{dp(x)}{dx}$	(def. pág. 27)	distribuição de pressão ao longo do poço
$K(x)$	(def. pág. 32)	resistividade entre os fluxos do reservatório e do poço
$p_{res}$	(def. pág. 32)	pressão no reservatório ( <i>far field</i> )
$x_{heel}$	(def. pág. 33)	coordenada relativa ao calcanhar do poço horizontal
$x_{toe}$	(def. pág. 33)	coordenada relativa ao dedão do poço horizontal
$p_{heel}$	(def. pág. 33)	pressão no calcanhar do poço horizontal
$Q_{toe}$	(def. pág. 33)	vazão no dedão do poço horizontal
$Q_w$	(def. pág. 34)	vazão total do poço vertical (equação poço vertical)
$k$	(def. pág. 34)	permeabilidade meio poroso (equação poço vertical)
$h$	(def. pág. 34)	altura do reservatório (equação poço vertical)
$p_w$	(def. pág. 34)	pressão no poço vertical (equação poço vertical)
$r_{res}$	(def. pág. 34)	raio externo do reservatório (equação do poço vertical)
$r_w$	(def. pág. 34)	raio do poço vertical (equação do poço vertical)
$q_{lw}$	(def. pág. 34)	fluxo por unidade de comprimento (poço vertical)
$K_{vw}$	(def. pág. 34)	resistividade para o poço vertical
$\bar{K}_{vw}$	(def. pág. 35)	resistividade adimensional para o poço vertical

$l_{hw}$	(def. pág. 35)	comprimento total do poço horizontal
$B_{res}$	(def. pág. 35)	largura do reservatório
$H_{res}$	(def. pág. 35)	altura do reservatório
$\alpha(x)$	(def. pág. 35)	fator de correção
$K_{hw}$	(def. pág. 35)	resistividade constante para o poço horizontal
$\bar{K}(x)$	(def. pág. 36)	resistividade adimensional para o poço horizontal
$\omega(x)$	(def. pág. 36)	função peso do polinômio de <i>Legendre</i>
$\bar{K}_p(x)$	(def. pág. 36)	resistividade adimensional aproximada por polinômios
$a_n$	(def. pág. 36)	$n$ -ésimo coeficiente do $n$ -ésimo polinômio de <i>Legendre</i>
$L_n$	(def. pág. 36)	$n$ -ésimo polinômio de <i>Legendre</i>
$E_7$	(def. pág. 38)	função objetivo da equação de quadrados mínimos
$\vec{a}$	(def. pág. 38)	vetor de saída esperado das <i>MLPs</i> ( $\equiv \vec{d}$ )
$\vec{x}$	(def. pág. 39)	vetor sinal de entrada das <i>MLPs</i>
$n_t$	(def. pág. 39)	número total de casos gerados para compor o conjunto $A$
$A$	(def. pág. 39)	conjunto que contém todos os casos para treinamento
$A_T$	(def. pág. 40)	conjunto de treinamento (ajustes dos pesos sinápticos)
$A_v$	(def. pág. 40)	conjunto de validação
$A_{te}$	(def. pág. 40)	conjunto de teste
$p_{toe}$	(def. pág. 43)	pressão no dedão do poço horizontal
$\Delta p$	(def. pág. 43)	diferencial de pressão, usado no <i>Runge-Kutta</i>
$m_i$	(def. pág. 43)	fatores do algoritmo de <i>Runge-Kutta</i>
$k$	(def. pág. 43)	ponto no domínio do poço para o algoritmo de <i>Runge-Kutta</i>
$L^2$	(def. pág. 44)	norma $L^2$
$E_f$	(def. pág. 44)	erro em norma $L^2$ para análise das curvas no poço
$L^1$	(def. pág. 45)	norma $L^1$
$E_Q$	(def. pág. 45)	erro em norma $L^2$ da curva $Q_{hw}(x)$
$E_{dQdx}$	(def. pág. 45)	erro em norma $L^2$ da curva $\frac{dQ_{hw}(x)}{dx}$
$E_p$	(def. pág. 45)	erro em norma $L^2$ da curva $p(x)$
$E_{dpdx}$	(def. pág. 45)	erro em norma $L^2$ da curva $\frac{dp(x)}{dx}$
$E_{Q_{hw}}^{L^1}$	(def. pág. 45)	erro em norma $L^1$ da vazão do poço $Q_{hw}$
$E_{Q_{hw}}$	(def. pág. 45)	erro em norma $L^1$ alterada da vazão do poço $Q_{hw}$
$\sigma$	(def. pág. 45)	desvio padrão amostral

# **Capítulo 1**

## **Introdução**

O uso de redes neurais artificiais em diversas áreas do conhecimento humano tem sido propagado nas últimas décadas devido aos grandes avanços no desenvolvimento de computadores, fato que reestimulou as pesquisas na área de Inteligência Artificial (IA) e suas ramificações.

O avanço na capacidade de processamento de dados também estimulou pesquisas e desenvolvimento de simuladores numéricos envolvendo problemas físicos mais complexos de forma a obter simulações mais realistas e datalhadas. Além do mais, a busca por simulações eficientes e rápidas, principalmente em aplicações industriais, tem estimulado a interação entre as duas áreas: redes neurais artificiais e simulação numérica.

O acoplamento poço-reservatório é um problema muito comum na engenharia de petróleo quando se trata de poços perfurados de forma horizontal (poços horizontais). Embora existam algumas equações analíticas e semi-analíticas que o descrevem, como apresentado no trabalho de Joshi (1991), o acoplamento poço-reservatório tem sido simulado usando métodos numéricos tais como elementos finitos, diferenças finitas, volumes finitos etc. Em geral, cada autor propõe metodologias diferentes para acoplar as formulações de fluxo no reservatório com as do poço, de acordo com as necessidades de cada um, mas todos procuram obter informações e detalhes sobre o comportamento dos fluxos no interior do poço.

Neste contexto, o presente trabalho tem por objetivo aplicar redes neurais artificiais na simulação numérica do acoplamento poço-reservatório, buscando uma metodologia que possa ser aplicada em outras áreas e problemas da engenharia.

### **1.1 Apresentação: Redes Neurais Artificiais**

As pesquisas em Inteligência Artificial (IA) nas últimas décadas proporcionaram o surgimento de diversos tipos e arquiteturas de redes neurais, cada uma com características próprias e aplicações

variadas. Um tipo de rede neural artificial muito comum é o *perceptron* de múltiplas camadas (*MLP - Multilayer Perceptron*). Este tipo de rede neural é amplamente utilizado para mapeamento de funções devido à sua capacidade de "aprender" dados existentes e de generalizar valores. Assim, apresentando um conjunto de padrões ou dados de treinamento para a *MLP*, após o processo de treinamento, a rede poderá, idealmente, ser capaz de generalizar valores relativos à função associada com os dados de treinamento. Em termos matemáticos, um conjunto de dados de treinamento  $A$  é definido por:

$$A = \{(\vec{x}_i, \vec{y}_i) : \vec{y}_i = f(\vec{x}_i), i = 1, \dots, n, n \in \mathbb{N}\},$$

em que  $f(\vec{x}_i)$  é uma relação definida sobre  $A$ .

A função  $f(\vec{x}_i)$  pode ser uma expressão analítica, um modelo numérico ou mesmo uma relação observada a partir de experimentos. Espera-se que a rede *MLP* treinada a partir de  $A$  gere valores  $\vec{v}_j$  tais que  $\vec{v}_j \cong \vec{y}_j = f(\vec{x}_j)$ ,  $j \in \mathbb{N}$ , para  $j \neq i$  e  $j = i$ , descrevendo assim a capacidade da *MLP* de mapear a função  $f(\vec{x}_i)$ .

O trabalho utilizou o modelo tridimensional de acoplamento poço-reservatório desenvolvido por Devloo *et al.* (2009) para gerar um conjunto de dados do tipo  $A$  para treinamento de uma *MLP*. É apresentada uma técnica desenvolvida para gerar o conjunto  $A$  de forma que a *MLP* treinada busque fornecer um parâmetro de fluxo do reservatório para um modelo simplificado unidimensional do problema poço-reservatório. A ideia é que esse modelo simplificado consiga gerar resultados semelhantes ao modelo tridimensional.

## 1.2 Motivação

Redes neurais artificiais podem ser aplicadas em diversos problemas físicos da engenharia, compondo desde controle e automação de processos até mapeamento de funções. O grupo de pesquisa do Laboratório de Mecânica Computacional - LabMeC, da Faculdade de Engenharia Civil, Arquitetura e Urbanismo da UNICAMP, tem interesse no domínio de tal tecnologia para uso nas pesquisas relacionadas à mecânica computacional e à simulação numérica. Foi definido como caso de uso um problema numérico de acoplamento poço-reservatório, objeto de projetos de pesquisa do laboratório e de interesse da indústria e engenharia de petróleo.

A motivação se faz em utilizar as redes neurais juntamente com um modelo simplificado de forma a substituir o modelo de elementos finitos quando do estudo prévio dos parâmetros que regem o problema físico. Explica-se: o modelo de elementos finitos demanda um certo tempo para simular as equações tridimensionais dos fluxos; porém, para um estudo prévio de projeto de poço

ou mesmo para entender o comportamento de fluxos quando se alteram os parâmetros, um tempo longo é demandado para obter todas as respostas. Para agilizar esse processo de projeto prévio e do estudo do comportamento do problema físico, surgiu a ideia de se utilizar um modelo substituto que pudesse gerar resultados semelhantes ao do modelo original, porém em um tempo extremamente curto. O modelo substituto deveria ser capaz de interagir com o projetista ou engenheiro de forma que esse poderia variar um parâmetro qualquer, por meio de uma barra de rolagem, por exemplo, e o modelo geraria os resultados instantaneamente. Diante disso, optou-se em utilizar a tecnologia de redes neurais artificiais como composição desse modelo substituto.

### 1.3 Objetivos

A teoria de redes neurais artificiais vem se consolidando, mundialmente, como uma nova e eficiente ferramenta para lidar com a ampla classe dos assim chamados problemas complexos, em que extensas massas de dados devem ser modelados e analisados em um contexto multidisciplinar, envolvendo, simultaneamente, tanto os aspectos estatísticos e computacionais como os dinâmicos e de otimização (Kovács, 2006). Em vista disto, o uso de redes neurais artificiais nos mais diversos problemas da engenharia moderna não seria nada mais do que um pequeno passo neste processo de evolução de ferramentas numéricas associadas com inteligência artificial e teorias conexionistas.

Nesse contexto, o trabalho tem por objetivo justamente estudar, desenvolver e utilizar das características de uma rede neural do tipo *perceptron* de múltiplas camadas (*MLP*) num problema de simulação numérica da engenharia de petróleo: o acoplamento poço-reservatório. Por se tratar de um problema complexo, o trabalho visará analisar o uso da *MLP* num modelo numérico já desenvolvido do acoplamento (Devloo *et al.*, 2009), observando as potencialidades e limitações da rede neural no presente tema. Além do mais, a metodologia desenvolvida não tem a intenção de se restringir a esse problema, mas sim de deixar potencialidade de aplicação em outros campos da engenharia.

### 1.4 Organização do Trabalho

O trabalho é organizado em 4 capítulos e um apêndice, a saber: Revisão Bibliográfica, Metodologia, Resultados, Conclusão e Apêndice A. A seguir, são descritos resumidamente o conteúdo de cada um.

Em Revisão Bibliográfica são apresentados conceitos sobre redes neurais artificiais e sobre o modelo de acoplamento poço-reservatório. Na seção sobre redes neurais, descreve-se o neurônio artificial, algumas arquiteturas de redes e o processo de aprendizagem utilizado no trabalho

(retropropagação). O acomplamento poço-reservatório é descrito de acordo com o modelo tridimensional desenvolvido por Devloo *et al.* (2009). São descritas as equações de escoamento no reservatório e no poço, assim como as curvas de fluxo no interior do poço.

No capítulo Metodologia, são descritas as ferramentas computacionais utilizadas, o modelo simplificado unidimensional do acomplamento poço-reservatório, a análise de um parâmetro de fluxo (resistividade  $K(x)$ ), o processo de aprendizagem das redes neurais geradas (*MLPs*) e o arranjo das *MLPs* com o modelo unidimensional.

O capítulo Resultados apresenta alguns exemplos de validação da biblioteca produzida para geração de redes neurais (*NeuralLib*). São apresentados também os resultados referentes à aprendizagem das *MLPs* utilizadas e os resultados referentes ao arranjo do modelo unidimensional com essas *MLPs*. Gráficos e tabelas comparativas foram produzidas como forma de analisar a aprendizagem de cada *MLP* e a resolução do modelo unidimensional.

No capítulo Conclusão são apresentados discussões sobre o trabalho como todo, ressaltando alguns detalhes observados em Resultados.

O Apêndice A descreve em maiores detalhes a biblioteca *NeuralLib*. São descritas as classes implementadas, um caso de uso para geração de uma rede tipo *MLP* e uma descrição sobre o formato do arquivo de treinamento.

# **Capítulo 2**

## **Revisão Bibliográfica**

Procurou-se elencar os princípios e conceitos fundamentais dos dois assuntos-chave deste trabalho: redes neurais artificiais e acomplamento poço-reservatório. A revisão bibliográfica descreve cada assunto num âmbito mais generalista e detalha os tópicos que de fato são empregados no presente trabalho.

Alguns exemplos de aplicação de redes neurais artificiais como ferramenta auxiliar em modelagens numéricas relacionados à indústria do petróleo podem ser encontrados nos trabalhos de Aminzadeh *et al.* (2000), Nguyen *et al.* (2002), Alrumah (2003), Pimentel *et al.* (2005) e Villanueva (2008).

### **2.1 Redes Neurais Artificiais**

Descrever-se-á brevemente sobre redes neurais artificiais, contextualizando os princípios fundamentais que regem seu funcionamento, potencialidades e tipologias de redes. Inicialmente explicar-se-á a unidade básica de processamento à semelhança do cérebro humano: o neurônio artificial. Formas de arranjo de tais neurônios (arquitetura de redes) e tipos de treinamento serão descritos em sequência. Um algoritmo de aprendizagem também é descrito ao decorrer da seção. A revisão é baseada principalmente nos trabalhos de Haykin (2001), de Braga *et al.* (2007) e de Zuben (1996). Outros trabalhos também serviram como base teórica nesse assunto: Fausett (1994), Hassoun (1995) e Galushkin (2007).

#### **2.1.1 Definição**

Redes neurais artificiais, bem conhecidas também como redes neurais, são modelos matemáticos que tentam descrever o funcionamento de um cérebro quanto ao processamento de informações.

O cérebro é um computador (sistema de processamento de informação) altamente complexo, não-linear e paralelo, capaz de organizar seus constituintes estruturais, conhecidos como neurônios, de forma que processamentos complexos sejam realizados numa velocidade muito maior do que os computadores digitais atualmente existentes. Essa capacidade do cérebro de realizar tarefas complexas, apenas usando unidades simples de processamento (neurônios), motiva os estudos e avanços sobre redes neurais artificiais na tentativa de modelar tal capacidade utilizando-se componentes eletrônicos ou mesmo programação em computadores digitais.

### 2.1.2 Neurônio Artificial

Os neurônios artificiais são unidades de processamento de informação fundamentais para o funcionamento das redes neurais artificiais. Tais unidades de processamento nada mais são que modelos artificiais de um neurônio biológico. Assim, os avanços nos estudos sobre o funcionamento do cérebro e, consequentemente, dos neurônios biológicos, ocorridos nas primeiras décadas do século passado têm motivado fisiologistas, físicos e matemáticos a encontrarem um modelo que assim descrevesse seus comportamentos. Uma breve descrição sobre o funcionamento do neurônio biológico é realizada na sequência.

#### Neurônio Biológico

Os neurônios biológicos, segundo Kovács (1997), assim como qualquer célula biológica, são delimitados por uma fina membrana celular a qual, além de sua função biológica normal, possui determinadas propriedades que são essenciais para o funcionamento elétrico dessas células nervosas.

Estruturalmente, os neurônios biológicos são divididos, de forma geral, em três seções: o corpo celular, os dendritos e o axônio. Cada seção possui funções específicas que, em conjunto, caracterizam o funcionamento do neurônio. O corpo celular apresenta-se como a seção de menor dimensão, da ordem de milésimos de milímetro. Os dendritos são estruturas com alguns milímetros, e o axônio já pode ser mais longo. Um neurônio pode possuir muitos dendritos, mas possui somente um axônio. Basicamente, os dendritos têm a função de receber sinais elétricos provindos de outros neurônios, de forma que o corpo celular, ao processar tais sinais, envia-os para outros neurônios através do axônio. Assim, o fluxo de sinais elétricos num neurônio inicia pela recepção nos dentritos, sendo processado no corpo celular e transmitido pelo axônio para outros dentritos de outros neurônios, reinicializando o ciclo. O ponto de contato entre o axônio de um neurônio e o dentírito de outro é conhecido como sinapse, e é através das sinapses que os neurônios se unem funcionalmente, compondo as redes neurais biológicas.

O tipo mais comum de sinapse é a sinapse química. O processo subjacente consiste na conversão de um sinal elétrico provindo de um axônio (pré-sináptico) em um sinal químico o qual, por sua vez, é convertido num outro sinal elétrico (pós-sináptico), que é transmitido para o dendrito de outro neurônio. Assume-se que a sinapse é uma conexão simples que pode impor ao neurônio receptivo excitação ou inibição, mas nunca as duas. Os sinais pós-sinápticos provindos dos diversos dendritos são combinados no corpo celular, o qual, dependendo do percentual de excitação, emite um impulso através do axônio para os neurônios seguintes. Este percentual de excitação (ou potencial de ação) é conhecido como limiar de disparo (Kovács, 1997), e separa o estado do neurônio em repouso ou em disparo. Após a geração de um impulso (disparo), o neurônio entra em um período de refração absoluta (na qual fica incapaz de emitir outro sinal), seguido de um período de refração relativa, correspondente a uma elevação no limiar de disparo (Kovács, 1997).

### Modelos de um Neurônio

O primeiro modelo de um neurônio biológico foi proposto pelo fisiologista *Warren McCulloch* juntamente com *Walter Pitts* (McCulloch e Pitts, 1943), em 1943, e tratava-se de um modelo simples que descrevia um funcionamento booleano da célula (Kovács, 1997). Este modelo consistia em  $m$  terminais de entrada (dendritos) que recebem  $m$  valores  $x_1, x_2, \dots, x_m$  (que representam as ativações dos neurônios anteriores) e apenas um terminal de saída  $y$  (representando o axônio). As sinapses foram representadas por pesos  $w_1, w_2, \dots, w_m$ , acoplados nos terminais de entrada. Tais pesos podem possuir valores positivos ou negativos. Assim, o efeito da sinapse de um neurônio  $i$  pré-sináptico que emite um sinal  $x_i$  é  $w_i x_i$ . O percentual de excitação de tal modelo é feito somando todas as sinapses correspondentes  $\sum w_i x_i$ . O disparo ou não do neurônio então se dá comparando o resultado dessa soma com o limiar de ativação, a partir de uma função de ativação. Por se tratar de um modelo binário, a saída do neurônio de *McCulloch* e *Pitts* pode ser pulso (1) ou não-pulso (0).

O diagrama abaixo ilustra o modelo proposto de *McCulloch* e *Pitts*:

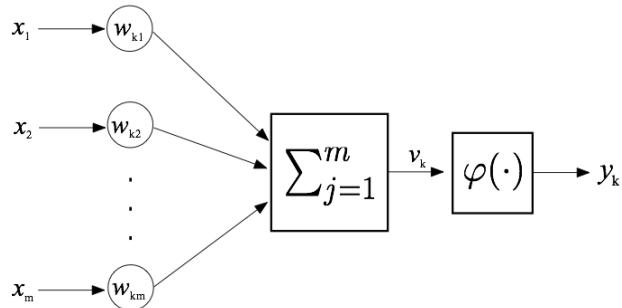


Figura 2.1: Modelo simplificado do neurônio de *McCulloch* e *Pitts*.

A função de ativação, também conhecida como função restritiva, tem por objetivo limitar a amplitude do sinal de saída do neurônio. Para o modelo de *McCulloch e Pitts*, a função de ativação é a função de limiar, representada matematicamente como:

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases}. \quad (2.1)$$

O modelo de *McCulloch e Pitts* pode ser esquematizado em uma forma mais geral de forma a possibilitar diversas aplicações: pode-se inserir um polarizador ou *bias* na entrada do neurônio e alterar a função de ativação  $\varphi(\cdot)$ , utilizando para isso outras funções. O *bias* tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação. Quatro elementos básicos podem ser identificados nesse modelo mais geral, a saber:

1. Sinapses.
2. Somador.
3. Função de Ativação.
4. *Bias* (polarizador).

O funcionamento do neurônio da Figura 2.1, acrescido do *bias*, pode ser descrito de forma matemática como:

$$u_k = \sum_{j=1}^m w_{kj}x_j \quad (2.2)$$

e

$$y_k = \varphi(u_k + b_k) \quad (2.3)$$

em que  $x_1, x_2, \dots, x_m$  são sinais de entrada;  $w_{k1}, w_{k2}, \dots, w_{km}$  são os pesos sinápticos do neurônio  $k$ ;  $u_k$  é a saída do combinador linear;  $b_k$  é o *bias* ou polarizador;  $\varphi(\cdot)$  é a função de ativação;  $y_k$  é o sinal de saída do neurônio. O termo  $u_k + b_k$  é chamado também de campo local induzido ou potencial de ativação, definido como:

$$v_k = u_k + b_k. \quad (2.4)$$

A função de ativação estipula a saída do neurônio em função do campo local induzido, e pode apresentar diversas topologias de acordo com as funções que a definem. Em geral, as funções de ativação restrigem a amplitude de saída do neurônio em um intervalo unitário fechado  $[0, 1]$  ou

$[-1, 1]$ . Abaixo seguem algumas funções de ativação apresentadas em Haykin (2001) e em Braga *et al.* (2007):

- Função de Limiar. Esta função apresenta a seguinte forma matemática:

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases}. \quad (2.5)$$

A Figura 2.2 ilustra o gráfico dessa função:

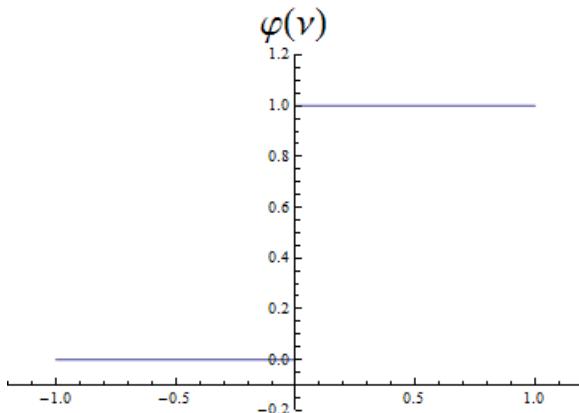


Figura 2.2: Função de limiar.

- Função Linear por Partes. Tal função, cujo gráfico é apresentado na Figura 2.3, pode ser descrita como:

$$\varphi(v) = \begin{cases} 1, & v \geq +1/2 \\ v + 1/2, & +1/2 > v > -1/2 \\ 0, & v \leq -1/2 \end{cases}. \quad (2.6)$$

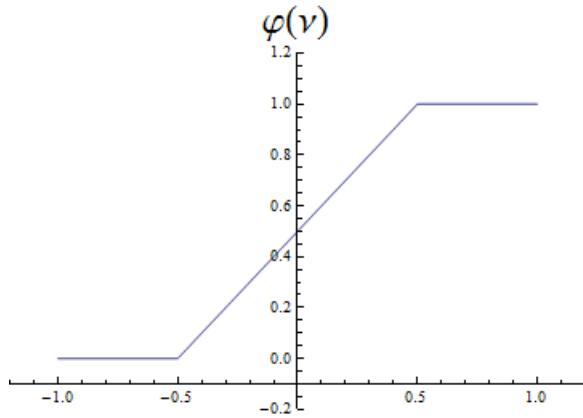


Figura 2.3: Função linear por partes.

- Função Sigmóide. Esta função apresenta um gráfico contínuo e suave, sendo também diferenciável. A função sigmoidal pode ser representada pela função logística ou pela função tangente hiperbólica. Fundamentalmente, a diferença entre elas está nos intervalos nos quais são delimitadas. As Figuras 2.4 e 2.5 ilustram os gráficos dessas funções. As expressões da função logística e da função tangente hiperbólica são representadas por 2.7 e 2.8, respectivamente:

$$\varphi_l(v) = a \cdot \frac{1}{1 + \exp(-bv)} \quad (2.7)$$

$$\varphi_{th}(v) = a \cdot \frac{\exp(bv) - \exp(-bv)}{\exp(bv) + \exp(-bv)}. \quad (2.8)$$

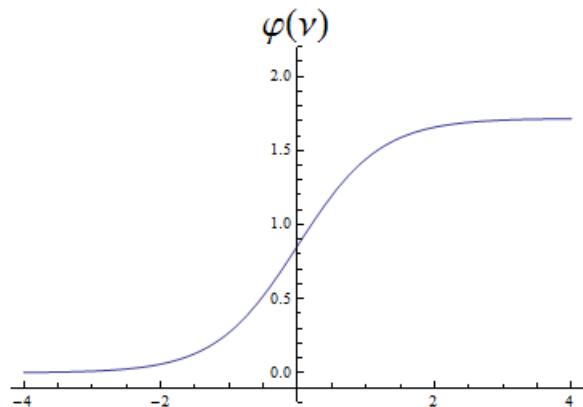


Figura 2.4: Função logística.

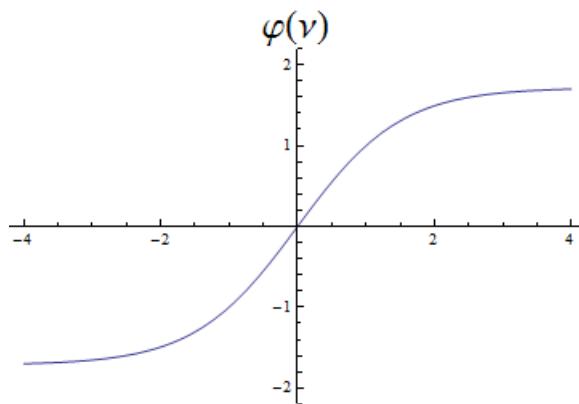
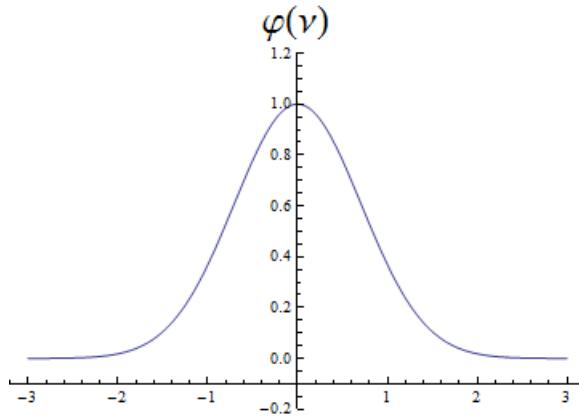


Figura 2.5: Função tangente hiperbólica.

- Função de Base Radial. Um exemplo deste tipo de função é a função *Gaussiana*. A Figura 2.6 ilustra o gráfico gerado por 2.9:

$$\varphi(v) = \exp\left(-\frac{(v-u)^2}{r^2}\right) \quad (2.9)$$

Figura 2.6: Exemplo de função de base radial: função *Gaussiana*.

em que  $u$  é o ponto no qual a função está centrada e  $r$  é dispersão da função ao longo do eixo  $v$ . Na Figura 2.6,  $u = 0$  e  $r = 1$ .

### 2.1.3 Arquiteturas de Redes

A arquitetura de uma rede neural é a maneira como seus neurônios estão organizados. Essa organização entre neurônios está intimamente ligada com o algoritmo de aprendizagem usado para treinamento da rede. De forma geral, Haykin (2001) identifica três classes de arquiteturas diferentes:

#### Redes Alimentadas Adiante com Camada Única

Uma forma de organização de redes é em camadas, ou seja, os neurônios se estruturam na forma de camadas. Na forma mais simples de uma rede em camadas, tem-se uma camada de entrada de nós de fonte que se projeta sobre uma camada de saída de neurônios, mas não vice-versa. Este tipo de arquitetura é estritamente do tipo alimentada adiante ou acíclica. O fato de ser referida como rede de camada única é que somente a camada de saída realiza "cálculo" sobre os dados, ou seja, a camada de entrada somente "reproduz" os sinais de entrada para cada neurônio da camada de saída. A Figura 2.7 ilustra esse tipo de arquitetura:

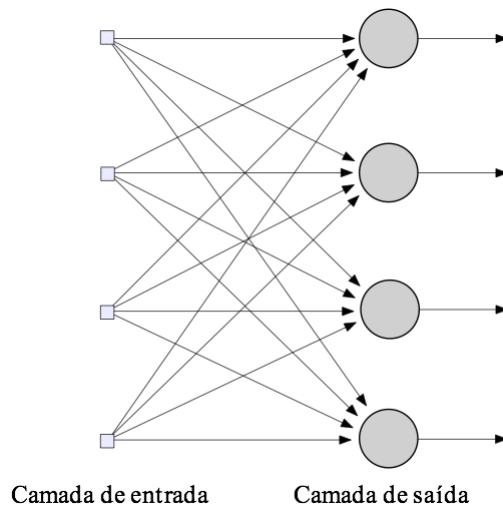


Figura 2.7: Rede alimentada adiante com uma camada de entrada e uma de saída.

#### Redes Alimentadas Diretamente com Múltiplas Camadas

Este tipo de arquitetura tem a característica de apresentar uma ou mais camadas intermediárias entre as camadas de entrada e de saída. Essas camadas intermediárias são referidas como ocultas, nas quais os neurônios nelas presentes são referenciados como neurônios ocultos. A função dos

neurônios ocultos é intervir entre a entrada externa e a saída da rede de uma maneira útil. Neste arranjo de estrutura, os nós da camada de entrada distribuem o vetor de entradas (sinais de entrada da rede neural) para os neurônios da segunda camada. Os sinais de saída da segunda camada são passados para a terceira camada como sinais de entrada, e assim se repetirá até a camada de saída da rede neural, sendo que o conjunto de sinais dos neurônios desta última camada constituirá a resposta global da rede para o padrão de ativação fornecido pela camada de entrada. A Figura 2.8 ilustra uma rede com uma camada oculta:

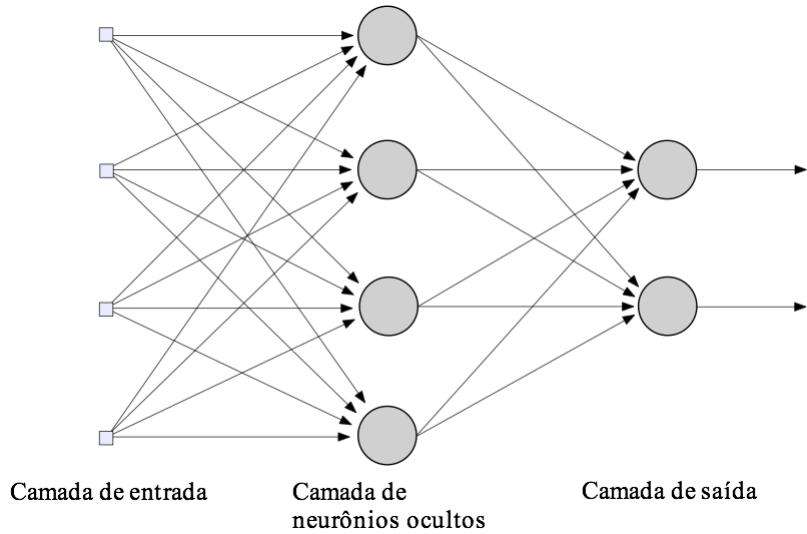


Figura 2.8: Rede alimentada adiante com uma camada oculta de neurônios.

## Redes Recorrentes

Uma rede recorrente se distingue de uma rede neural alimentada adiante por ter pelo menos um laço de realimentação. Uma rede desse tipo pode apresentar neurônios ocultos ou não. A presença de laços de realimentação tem um impacto profundo na capacidade de aprendizagem da rede e no seu desempenho. Somado a isso, esses laços de realimentação envolvem o uso de ramos particulares compostos de elementos de atraso unitário, os quais realimentam alguns neurônios da rede com saídas processadas no passo de tempo anterior. Essas realimentações conferem à rede um comportamento dinâmico não-linear. As figuras 2.9 e 2.10 mostram duas redes distintas, mas que apresentam pelo menos um laço de realimentação:

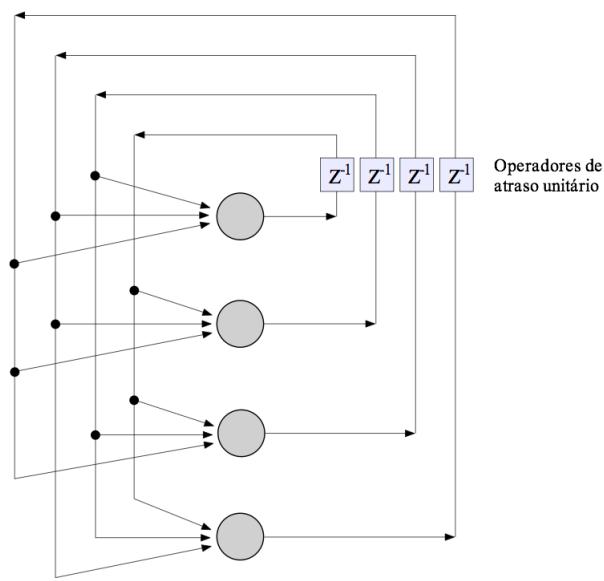


Figura 2.9: Rede recorrente autoalimentada.

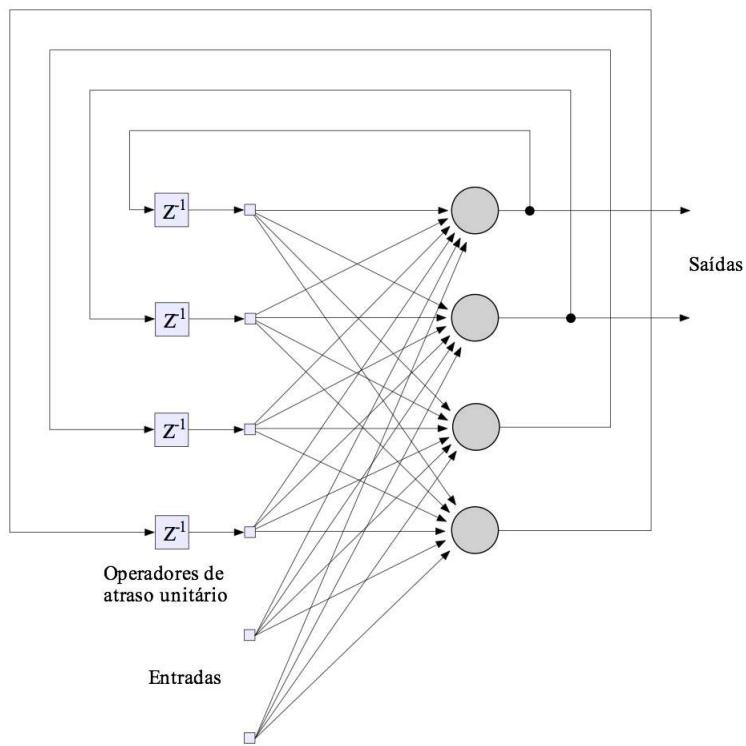


Figura 2.10: Rede recorrente com entrada e saída.

### 2.1.4 Processos de Aprendizagem

A habilidade de aprender a partir de seu ambiente é uma propriedade fundamental para uma rede neural. O processo de aprendizagem destaca, além disso, um papel importante para melhoria de desempenho da rede segundo algum critério preestabelecido. Em geral, uma rede neural aprende acerca de seu ambiente a partir de um processo de ajustes sobre seus pesos sinápticos e *biases*. Idealmente, a rede se torna mais instruída sobre o seu ambiente após cada iteração do processo de aprendizagem (Haykin, 2001).

Dentre diversas definições ou atividades relacionadas ao conceito de aprendizagem como todo, é apresentada uma definição contextualizada no âmbito das redes neurais, conforme descrito em Haykin (2001):

*Aprendizagem é um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. O tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros ocorre.*

Segundo Braga *et al.* (2007), os processos de aprendizagem podem ser agrupados em dois paradigmas principais: aprendizado supervisionado e aprendizado não-supervisionado. Aprendizado supervisionado implica a existência de um supervisor ou professor, o qual é responsável por estimular as entradas da rede por meio de padrões de entrada e observar a saída da rede calculada pela mesma, comparando-a com a saída desejada. Esse tipo de aprendizado é aplicado em problemas em que se deseja mapear padrões de entrada e saída. No aprendizado não-supervisionado não existe a figura do supervisor ou professor e, em geral, apenas os padrões de entrada estão disponíveis. O processo de aprendizagem se dá pela existência de regularidades e redundâncias nos padrões de entrada apresentados à rede. Esse processo se aplica em problemas que visam à descoberta de características estatísticas nos dados.

Pode-se listar alguns processos de aprendizagem, elencando-os dentro desses dois paradigmas, de acordo com Braga *et al.* (2007):

#### **Aprendizado supervisionado:**

- Aprendizagem por correção de erro.
- Aprendizagem por reforço.

### Aprendizado não supervisionado:

- Aprendizagem *Hebbiana*.
- Aprendizagem competitiva.

Maiores detalhes sobre cada processo, ou mesmo sobre outros, podem ser encontrados em Fausett (1994), Hassoun (1995), Haykin (2001), Braga *et al.* (2007) e Galushkin (2007).

Os processos de aprendizagem podem, em geral, serem compostos por diferentes procedimentos ou algoritmos de aprendizagem. Algoritmo de aprendizagem pode ser definido como conjunto de regras bem-definidas utilizadas para resolução de um problema de aprendizagem (Haykin, 2001). Diversos algoritmos existem e, em geral, diferem um do outro pela forma como ajustam os pesos sinápticos dos neurônios. No presente trabalho, foi utilizado um algoritmo conhecido como retropropagação ou *backpropagation*, o qual faz parte do processo de aprendizado supervisionado, mais especificamente por correção de erro. Serão descritos, na sequência, o processo de aprendizagem por correção de erro e o algoritmo de retropropagação ou *backpropagation*.

### Aprendizagem por correção de erro

Este processo baseia-se em alterar os valores dos pesos sinápticos dos neurônios em função da diferença entre os sinais de saída da rede  $\vec{y}(n, k) = (y_1(n, k), \dots, y_i(n, k), \dots, y_m(n, k))$  e os valores esperados  $\vec{d}(k) = (d_1(k), \dots, d_i(k), \dots, d_m(k))$ , em que o índice  $n$  indica a  $n$ -ésima iteração no processo de aprendizagem,  $k$  é a  $k$ -ésima amostra do conjunto de treinamento e  $m$  é o número de neurônios na camada de saída. Essa diferença recebe o nome de sinal de erro  $\vec{e}(n, k) = (e_1(n, k), \dots, e_i(n, k), \dots, e_m(n, k))$ , definido como:

$$\vec{e}(n, k) = \vec{d}(k) - \vec{y}(n, k). \quad (2.10)$$

De forma geral, o sinal de erro  $\vec{e}(n, k)$  aciona um mecanismo de controle, cujo propósito é aplicar uma sequência de ajustes corretivos aos pesos sinápticos de cada neurônio da camada de saída e também das camadas ocultas. Os ajustes corretivos são projetados para aproximar passo a passo o sinal de saída  $\vec{y}(n, k)$  da resposta desejada  $\vec{d}(k)$ . Isto é realizado através da minimização de uma função de custo  $\xi(n, k)$ , definida como:

$$\xi(n, k) = \frac{1}{2} (\vec{e}(n, k) \cdot \vec{e}(n, k)) \quad (2.11)$$

sendo  $\xi(n, k)$  o valor instantâneo da energia do erro para a  $n$ -ésima iteração ou época, para a  $k$ -ésima amostra do conjunto de treinamento. Destaca-se que para contabilizar uma iteração ou época é necessário que todas as amostras (ou casos) do conjunto de treinamento sejam apresentadas à

rede para o processo de treinamento. Assim, na  $n$ -ésima iteração ou época, todas as amostras do conjunto de treinamento foram apresentadas pelo menos  $n - 1$  vezes à rede; ao final da  $n$ -ésima iteração, todas as amostras foram apresentadas  $n$  vezes.

A energia média do erro quadrado é obtida somando-se os  $\xi(n, k)$  de todo o conjunto de treinamento. Considerando  $T$  como sendo a dimensão da amostra de treinamento, tem-se a energia média do erro quadrado para  $n$ -ésima iteração ou época:

$$\xi_{med}(n) = \frac{1}{T} \sum_{k=1}^T \xi(n, k). \quad (2.12)$$

Os ajustes passo a passo dos pesos sinápticos de cada neurônio continuam até o sistema atingir um estado estável, isto é, os pesos sinápticos estão de tal forma estabilizados que o valor da energia do erro situa-se num ponto de mínimo (local ou global) na superfície de erro. Idealmente, o processo de treinamento deveria se encerrar quando a energia do erro atingisse seu mínimo global. Porém, isso nem sempre é possível devido às não-linearidades presentes na rede. Apesar disso, existem técnicas de treinamento que buscam levar a rede a se aproximar do ponto de mínimo global.

A minimização da função de custo  $\xi(n, k)$  resulta na regra de aprendizagem normalmente referida como regra delta ou regra de *Widrow-Hoff*. A minimização da função de custo é dependente do processo ou algoritmo de aprendizado utilizado. Na sequência, será descrito maiores detalhes dessa minimização, de acordo com o algoritmo de retropropagação ou *backpropagation*.

### Algoritmo de retropropagação (*backpropagation*)

A energia instantânea do erro  $\xi(n, k)$  e, consequentemente, a energia média do erro  $\xi_{med}(n)$ , são funções de todos os pesos sinápticos e *biases* dos neurônios da rede. Para um dado conjunto de treinamento,  $\xi_{med}(n)$  representa a função de custo como uma medida do desempenho de aprendizagem. O objetivo do processo de aprendizagem é ajustar os parâmetros livres (pesos sinápticos e *biases*) para minimizar  $\xi_{med}(n)$ . Os ajustes dos pesos são realizados de acordo com os respectivos erros calculados para cada padrão apresentado à rede. A média aritmética destas alterações individuais de peso sobre o conjunto de treinamento é, portanto, uma estimativa da alteração real que resultaria da modificação dos pesos baseada na minimização da função de custo  $\xi_{med}$  sobre o conjunto de treinamento inteiro. Será descrito a seguir a expressão de correção dos pesos sinápticos e a sequência de passos para realizá-la. Destaca-se que a expressão foi deduzida para uma amostra geral do conjunto de treinamento. Portanto, o índice  $k$  utilizado para referenciar a  $k$ -ésima amostra não será utilizado para esse fim.

A correção  $\Delta w_{ji}(n)$  aplicada ao peso sináptico  $w_{ji}(n)$  é definida pela regra delta:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \xi(n)}{\partial w_{ji}(n)} \quad (2.13)$$

em que  $\eta$  é o parâmetro da taxa de aprendizagem e  $j$  representa o  $j$ -ésimo neurônio da camada de saída cujo  $i$ -ésimo peso sináptico está sendo corrigido. O sinal negativo indica a descida do gradiente no espaço de pesos. O ajuste do peso  $w_{ji}$  é realizado como:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n). \quad (2.14)$$

A Equação 2.13 pode ser escrita como:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad (2.15)$$

em que  $\delta_j(n)$  é chamado gradiente local e  $y_i(n)$  é o sinal de saída do  $i$ -ésimo neurônio pré-sináptico ao neurônio  $j$ . O gradiente local  $\delta_j(n)$  é definido como:

$$\delta_j(n) = e_j(n) \varphi'_j(v_j(n)). \quad (2.16)$$

em que  $e_j(n)$  é o sinal de erro do neurônio  $j$  e  $\varphi'_j(v_j(n))$  é a derivada da função de ativação em relação à  $v_j(n)$ . Maiores detalhes sobre a definição do gradiente local são encontrados em Haykin (2001).

O gradiente local é um estimador que aponta para uma possível direção de modificações necessárias a serem efetuadas sobre os pesos sinápticos. De acordo com a Equação 2.16, o gradiente local  $\delta_j(n)$  para o neurônio de saída  $j$  é igual ao produto do sinal de erro  $e_j(n)$  com derivada  $\varphi'_j(v_j(n))$  da função de ativação associada a esse neurônio.

Nota-se pelas Equações 2.15 e 2.16 que um fator chave envolvido no cálculo do ajuste de peso  $\Delta w_{ji}(n)$  é o sinal de erro  $e_j(n)$  na saída do neurônio  $j$ . Neste contexto, identificam-se dois casos distintos dependendo de onde o neurônio  $j$  está localizado. No caso 1, o neurônio  $j$  é um nó de saída. No caso 2, o neurônio  $j$  é um nó oculto. Para cada caso, o cálculo de  $\Delta w_{ji}(n)$  assume formas diferentes, as quais são descritas abaixo:

1. Caso 1. Quando o neurônio  $j$  está localizado na camada de saída da rede, ele é suprido com uma resposta desejada particular. Podemos utilizar a Equação 2.10 para calcular o sinal de erro  $e_j(n)$  associado a esse neurônio. Tendo-se determinado  $e_j(n)$ , calcula-se diretamente o gradiente local  $\delta_j(n)$ , usando a Equação 2.16.
2. Caso 2. Quando o neurônio  $j$  está localizado em uma camada oculta da rede, não existe uma resposta desejada para aquele neurônio. Sendo assim, para cálculo do gradiente local, é necessário retropropagar os sinais de erros através da rede. Haykin (2001) desenvolve

o cálculo do gradiente local  $\delta_j(n)$  partindo da função de custo definida na Equação 2.11, chegando em:

$$\delta_j(n) = \varphi'_j(v_j(n)) \sum_{k=1}^m \delta_k(n) w_{kj}(n) \quad (2.17)$$

em que  $k$  é o  $k$ -ésimo neurônio da camada subsequente à camada do neurônio  $j$  (ou seja,  $k$  são os neurônios que recebem o sinal de saída do neurônio  $j$ ) e  $m$  é o número total de neurônios presentes nessa camada (os quais estão ligados ao neurônio  $j$  por meio dos pesos  $w_{kj}(n)$ ).

O algoritmo de retropropagação é definido então como a sequência de dois passos principais: propagação e retropropagação. São descritos a seguir:

### Propagação

Na propagação, os pesos sinápticos se mantêm inalterados em toda a rede e os sinais funcionais são calculados individualmente, neurônio a neurônio. Ou seja, a fase de propagação começa desde a camada de entrada, passando pelas camadas ocultas (quando essas existem), e termina na camada de saída, onde são calculados os sinais de erro.

### Retropropagação

O passo de retropropagação começa na camada de saída, a partir da qual os ajustes dos pesos são enviados para as camadas ocultas através dos cálculos dos gradientes locais  $\delta_j(n)$  para cada neurônio oculto. Todos os pesos sinápticos dos neurônios são então ajustados de acordo com as expressões definidas pela regra delta.

Após término do passo de retropropagação, inicializa-se o passo de propagação, recomeçando assim o algoritmo. Esse processo recursivo permite que os pesos sinápticos sofram modificações de acordo com a regra delta. Destaca-se que o passo de propagação e retropropagação podem ser realizados amostra por amostra do conjunto de treinamento, e as expressões acima foram deduzidas a partir disso. Essa maneira de realizar os ajustes dos pesos sinápticos é conhecida como modo sequencial de aprendizagem. Uma outra maneira de aplicar o algoritmo é realizar o passo de propagação sobre todas as amostras do conjunto de treinamento e somente depois realizar o passo de retropropagação. Essa forma é conhecida como modo por lote de treinamento e apresenta poucas diferenças no cálculo da correção  $\Delta w_{ji}(n)$  em relação ao modo sequencial. Maiores detalhes sobre o modo por lote são encontrados em Haykin (2001).

Um termo que pode ser acrescentado na Equação 2.14 com intuito de aumentar a taxa de aprendizagem evitando, porém, instabilidades na convergência do método, é um termo de momento, apresentado na sequência:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) + \alpha w_{ji}(n-1) \quad (2.18)$$

em que  $\alpha$  é usualmente um número positivo chamado de constante de momento. A Equação 2.18 é conhecida como regra delta generalizada.

## 2.2 Acoplamento Poço-Reservatório

O problema de acoplamento poço-reservatório é um assunto estudado durante décadas por diversos pesquisadores e profissionais da engenharia de petróleo, principalmente quando da simulação numérica do escoamento no meio poroso com o escoamento no poço horizontal. A diferença na natureza dos fluxos no reservatório e no poço requer o desenvolvimento de diversos modelos matemáticos que, a partir de algumas simplificações, apresentam-se na forma de uma equação analítica ou semi-analítica para tal problema. Porém, tais modelos analíticos e semi-analíticos podem não ser interessantes quando deseja-se ter mais detalhes do fluxo no interior do poço ou mesmo do reservatório. Desta forma, pesquisadores e profissionais recorrem aos simuladores numéricos de reservatórios, geralmente conhecidos como simuladores numéricos de fluxo (Rosa *et al.*, 2006), os quais apresentam maior complexidade em termos de modelagem matemática do problema físico. Assim, vários são os modelos de acoplamento poço-reservatório existentes, cada um concentrando-se em determinadas características e complexidades de acordo com as finalidades para as quais foram desenvolvidos. Pode-se listar os trabalhos de Gomes (1990), Lemos (1993), Dickstein *et al.* (1997), Vicente (2000), Arrieta (2004), Shirdel e Sepehrnoori (2009) e Devloo *et al.* (2009) como exemplos de modelos numéricos e os trabalhos de Joshi (1991) e, mais recentemente, de Escobar e Montealegre (2008) como exemplos de modelos analíticos e de correlações.

Os avanços no desenvolvimento e aperfeiçoamento na tecnologia de perfuração de poços permitiram que, nos últimos anos, uma grande quantidade de poços horizontais fosse perfurada no mundo (Arturo *et al.*, 2007; Vicente, 2000). De acordo com Fernandes *et al.* (2006), a Petrobras, desde a década de 90, vem desenvolvendo vários campos com poços horizontais para drenar desde carbonatos de baixa permeabilidade até arenitos altamente friáveis. Além disto, vários poços horizontais têm sido perfurados e completados para injeção de água, predominantemente em arenitos da Bacia de Campos (Fernandes *et al.*, 2006). O propósito dos poços horizontais é justamente aumentar o contato com o reservatório e, portanto, aumentar a produtividade do poço. As vazões para poços horizontais podem ser de 2 a 5 vezes maiores que para poços verticais não estimulados (Arturo *et al.*, 2007).

Devido ao crescente interesse prático, a tecnologia de poços horizontais tem recebido notável atenção e considerável quantidade de pesquisa. Segundo Vicente (2000), embora diversas fer-

ramentas numéricas e analíticas tenham sido desenvolvidas para investigar o comportamento do fluxo e predizer a performance de poços horizontais, as dificuldades inerentes de realizar essas tarefas desafiadoras são enormes. Assim, optou-se em utilizar no presente trabalho um modelo existente, desenvolvido por Devloo *et al.* (2009), e o qual apresenta notável grau de qualidade no quesito comportamento de fluxo no interior do poço. Descrever-se-ão as formulações e os métodos numéricos utilizados por Devloo *et al.* (2009), detalhando posteriormente os fluxos no reservatório e no poço e a maneira como foram acoplados. Outros detalhes sobre simulação numérica em engenharia de petróleo podem ser encontrados em Crichlow (1977) e em Aziz e Settari (1983). Uma visão geral sobre engenharia de petróleo pode ser encontrada no trabalho de Thomas (2001).

### 2.2.1 Modelo Tridimensional do Acoplamento Poço-Reservatório

O modelo utiliza o método de elementos finitos juntamente com a aplicação de algumas tecnologias avançadas, tais como elementos geométricos de mapeamentos curvos, refinamento direcional de malha, adaptatividade  $hp$ , redução dimensional e combinação dos métodos de elementos finitos tradicional e de Galerkin descontínuo.

O modelo assume escoamento monofásico isotérmico e implementa as equações de escoamento tanto no reservatório quanto no poço, as quais estão descritas em 2.2.2 e em 2.2.3. A geometria do reservatório contempla áreas de drenagem como sendo retangulares ou elípticas. As condições de contorno aplicadas são de fluxo nulo nas camadas confinantes (impermeáveis), pressão declarada na fronteira da área de drenagem e no calcanhar (*heel*) do poço horizontal. As permeabilidades horizontal e vertical do reservatório são consideradas de forma a contemplar sua anisotropia.

A malha desenvolvida baseou-se no mapeamento transfinito capaz de mapear exatamente o domínio do problema (Lucci, 2009). Desta forma, foram geradas malhas que representavam exatamente as geometrias dos poços e áreas de drenagem. O uso de refinamento direcional e seletividade de ordem de aproximação em cima dessas malhas permitiu melhorar a qualidade da solução sem comprometer o desempenho da simulação. As Figuras 2.11 e 2.12 esboçam o uso de elementos curvos para descrever a geometria e o detalhe dos refinamentos direcionais, respectivamente:

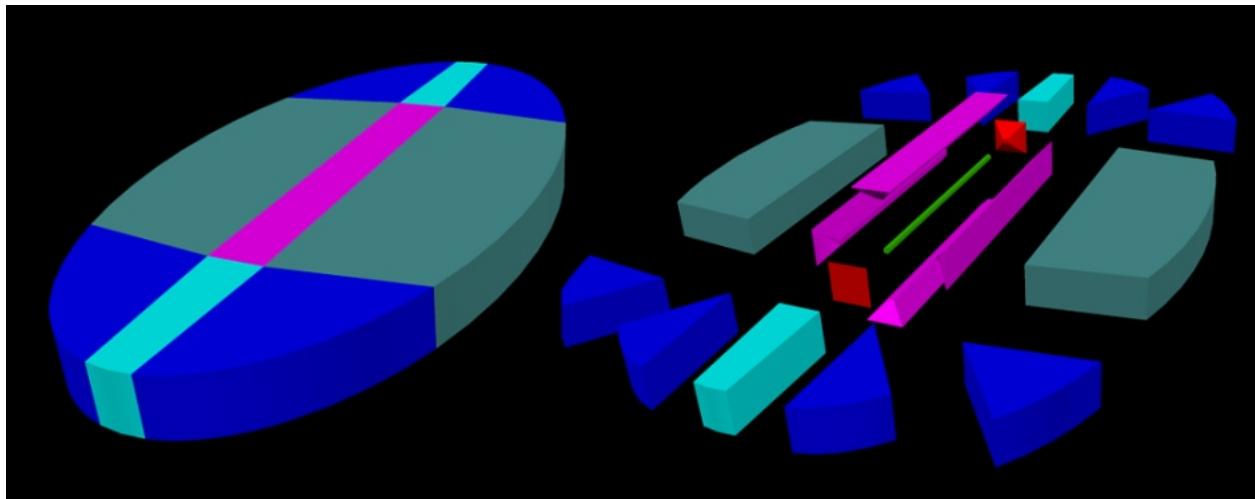


Figura 2.11: Poço horizontal e reservatório elíptico: elementos tridimensionais curvos.

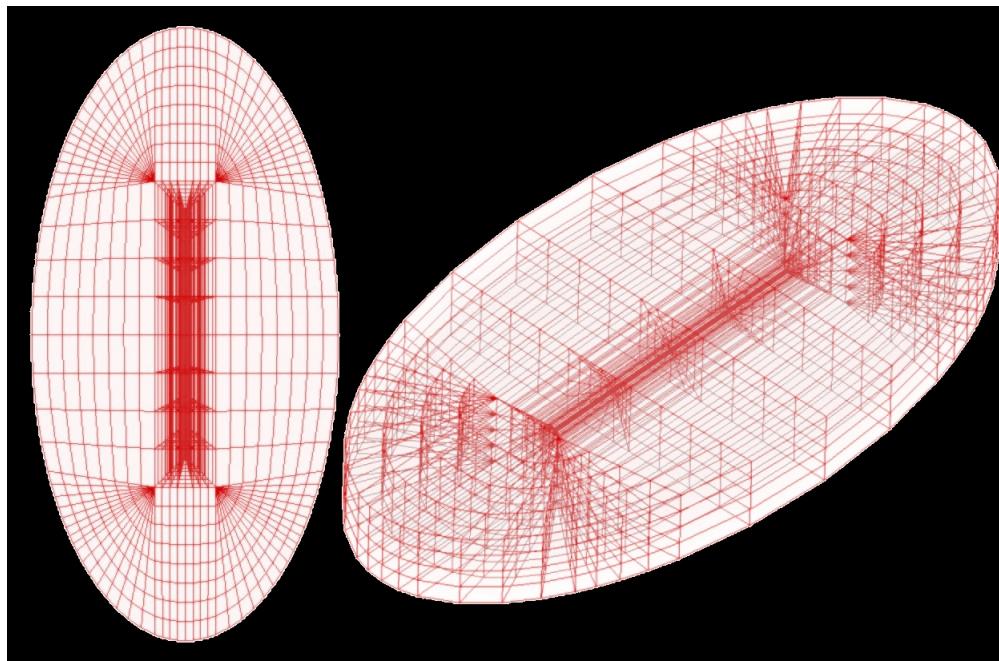


Figura 2.12: *Wireframe* da malha do poço horizontal e reservatório elíptico.

A formulação do escoamento no reservatório é linear. Já a formulação do escoamento no poço horizontal é não-linear. A resolução do problema acoplado poço-reservatório requer, portanto, a solução de um sistema de equações não-lineares. A sua resolução foi realizada através do método de

*Newton-Raphson* com busca unidimensional (*line search*) do tipo segmento áureo. O modelo assumiu o escoamento no poço horizontal como unidimensional. Assim, os elementos tridimensionais que descrevem o poço horizontal foram substituídos por elementos unidimensionais através da sistematica da redução dimensional, o que permitiu maior rapidez nas iterações de *Newton-Raphson*.

Os dados de entrada (*inputs*) para simulação no modelo tridimensional para poços horizontais são: geometria da área de drenagem do reservatório (retangular ou elíptico), comprimento do poço, raio de drenagem, afastamento vertical, permeabilidade horizontal, permeabilidade vertical, altura do reservatório, pressão no bordo do reservatório (*far field*), pressão no calcanhar (*heel*) do poço, diâmetro do poço, viscosidade do óleo, massa específica do óleo, fator *skin* ou de dano, opções de completação do poço ("poço aberto" ou tela/*liner*), dados do *liner* ou tela (quando a completação é do tipo tela/*liner*).

Nas próximas subseções, serão descritas as formulações para o escoamento no reservatório e no poço, assim como as curvas de fluxo resultantes do modelo.

### 2.2.2 Escoamento no Reservatório

O escoamento no reservatório é assumido como sendo monofásico e descrito pela lei de *Darcy*. A conservação de massa no reservatório, em regime permanente de fluido incompressível, implica em:

$$\operatorname{div} Q = 0 \quad (2.19)$$

com

$$Q = -\frac{1}{\mu} \overset{\leftrightarrow}{K} \nabla p \quad (2.20)$$

em que  $\overset{\leftrightarrow}{K}$  é o tensor de permeabilidade do meio poroso,  $\mu$  é a viscosidade do óleo e  $Q$  é o fluxo de fluido (vazão).

As condições de contorno tipo *Neumann* são definidas como

$$Q \cdot \vec{n} = -\frac{1}{\mu} \overset{\leftrightarrow}{K} \nabla p \cdot \vec{n} = \sigma_0 \quad (2.21)$$

isso é, o fluxo normal  $Q \cdot \vec{n}$  é um valor conhecido ( $\sigma_0$ ).

Podem-se aplicar, também, condições tipo *Dirichlet*, isto é, imposição de valores de pressões conhecidos  $p = p_0$ .

### 2.2.3 Escoamento no Poço Horizontal

O escoamento turbulento em tubos é descrito pela equação de *Fanning*, de acordo com Fernandes *et al.* (2006):

$$\frac{dp}{dx} = \frac{2fV^2\rho}{D} \quad (2.22)$$

em que  $V$  é a velocidade média do fluxo na seção circular do poço,  $\rho$  é a massa específica do fluido,  $D$  é o diâmetro do tubo e  $f$  é o fator de fricção, que é função do número de *Reynolds*.

Uma vez que a vazão no poço horizontal é descrita pela equação

$$Q_{hw} = V \frac{\pi D^2}{4} \quad (2.23)$$

tem-se que:

$$\frac{dp}{dx} = \frac{32f\rho Q_{hw}^2}{\pi^2 D^5} \quad (2.24)$$

ou

$$Q_{hw} = -\sqrt{\frac{\pi^2 D^5}{32f\rho} \frac{dp}{dx}}. \quad (2.25)$$

O sinal negativo indica que a vazão tem sentido do ponto de maior pressão para o de menor pressão. Pela lei de conservação de massa, tem-se que:

$$\frac{dQ_{hw}}{dx} + q_l = 0 \quad (2.26)$$

em que  $q_l$  é o fluxo de óleo entrando no poço.

O fator de fricção  $f$  é dado em função do número de *Reynolds* e do tipo de regime de escoamento. O número de *Reynolds* é um número adimensional, obtido pela expressão:

$$Reynolds = \frac{\rho V D}{\mu} \quad (2.27)$$

em que  $V$  é a velocidade média,  $\rho$  é a massa específica do fluido,  $D$  é o diâmetro do tubo e  $\mu$  a viscosidade do fluido.

Na literatura clássica, escoamentos com número de *Reynolds* inferiores a 2100 são considerados escoamentos laminares e, acima de 4000, turbulentos. Desta forma, para escoamentos laminares, o fator de fricção pode ser calculado como:

$$f_{laminar} = \frac{16}{Reynolds} \quad (2.28)$$

e, para escoamentos turbulentos (fórmula de *Blasius*), como:

$$f_{turbulento} = \frac{0,0791}{Reynolds^{0,25}}. \quad (2.29)$$

Uma vez que o escoamento em poços horizontais pode apresentar regimes diferentes, conforme a seção analisada do poço, poder-se-ia adotar o fator de fricção como sendo:

$$f = \begin{cases} f_{laminar}, & Reynolds \leq 2100 \\ f_{turbulento}, & Reynolds > 2100 \end{cases}. \quad (2.30)$$

Essa expressão, entretanto, conduziria a uma descontinuidade no valor do fator de fricção no ponto de *Reynolds* igual a 2100, o que seria bastante desfavorável para a qualidade do resultado da simulação numérica. Com a intenção de evitar esta inconsistência, Devloo *et al.* (2009) adotaram a seguinte expressão:

$$f = \begin{cases} f_{laminar}, & Reynolds \leq 1187,38 \\ f_{turbulento}, & Reynolds > 1187,38 \end{cases}, \quad (2.31)$$

que é contínua.

Substituindo as Equações 2.28 e 2.29 na Equação 2.31, e esta na Equação 2.25, tem-se:

$$Q_{hw} = \begin{cases} -\frac{\pi D^4}{128\mu} \frac{dp}{dx}, & Reynolds \leq 1187,38 \\ -\frac{2,252610888D^{19/7}}{\mu^{1/7}\rho^{3/7}} \left(\frac{dp}{dx}\right)^{4/7}, & Reynolds > 1187,38 \end{cases}. \quad (2.32)$$

Para composição do modelo, as Equações 2.19 e 2.32 são reescritas na forma fraca, de acordo com o método dos elementos finitos, e implementadas nos elementos específicos da malha: elementos de reservatório e elementos de poço, respectivamente.

## 2.2.4 Curvas de Fluxo do Modelo Tridimensional

Os resultados do modelo envolvem um valor numérico conhecido como IP (índice de produtividade), muito utilizado para verificar a viabilidade econômica do poço, o valor da vazão total  $Q_{hw}$ , correspondente à vazão no calcanhar (*heel*) do poço (ou seja,  $Q_{heel}$ ), e curvas de fluxo, geradas a partir do pós-processamento das soluções provenientes do cálculo de elementos finitos. As curvas mostram dados sobre o fluxo no interior do poço e são descritas na sequência:

1.  $p(x)$ . Esboça a pressão ao longo do eixo do poço (Figura 2.13);

2.  $Q_{hw}(x)$ . Esboça a vazão no interior do poço (Figura 2.14);
3.  $\frac{dQ_{hw}(x)}{dx}$ . Esboça a distribuição do fluxo ao longo do poço (Figura 2.15);
4.  $Rey(x)$ . Esboça o valor do número de *Reynolds* ao longo poço (Figura 2.16).

Ressalta-se que a coordenada  $x$  coincide com o eixo do poço. As figuras abaixo ilustram as curvas de fluxo geradas, nas quais a origem do sistema de coordenadas situa-se no *heel* do poço.

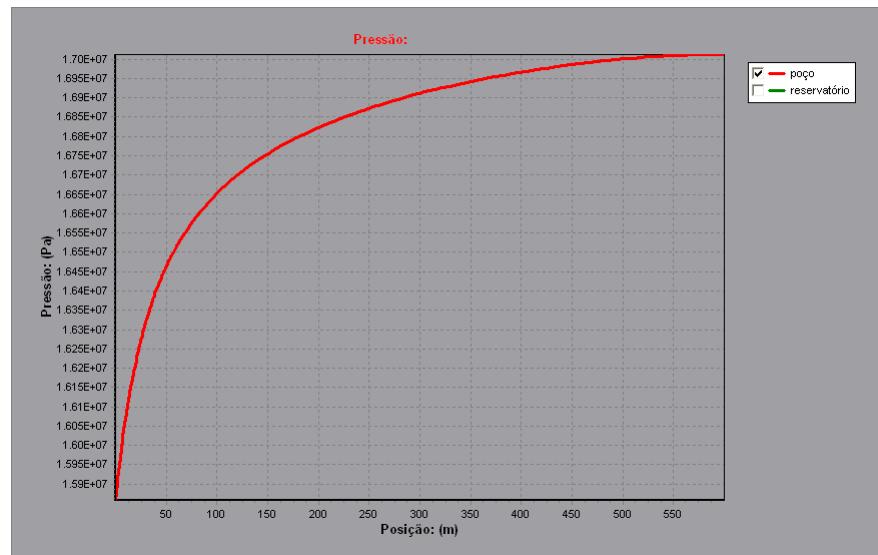


Figura 2.13: Curva de pressão no poço horizontal.

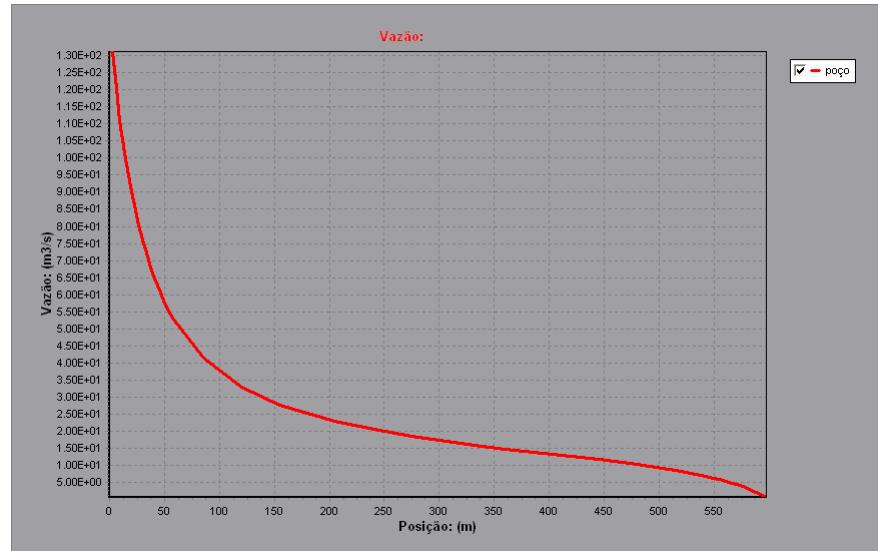


Figura 2.14: Curva de vazão no poço horizontal.

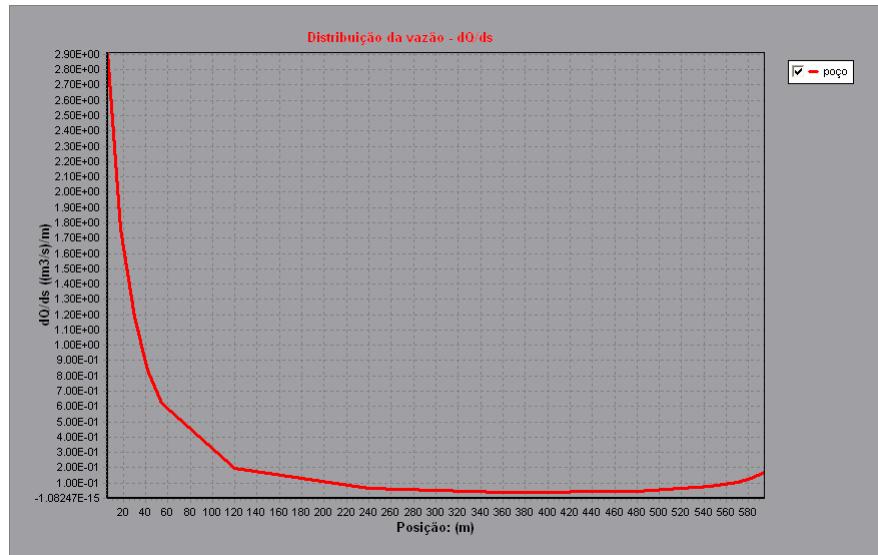


Figura 2.15: Curva de distribuição de vazão no poço horizontal.

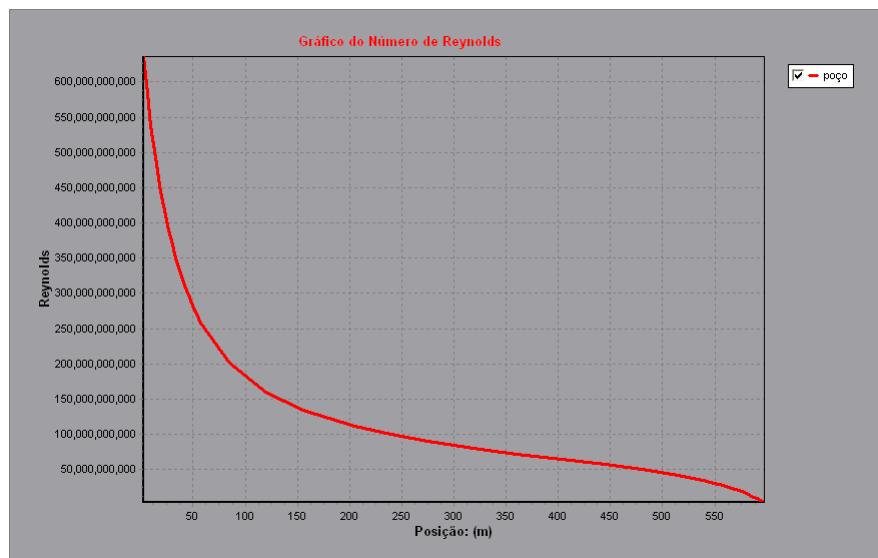


Figura 2.16: Valor do número de *Reynolds* no poço horizontal.

Por se tratar de um modelo numérico baseado no método dos elementos finitos, estas curvas não apresentam uma descrição analítica, mas são definidas a partir de pares de pontos (posição  $x$  no

poço, dado numérico de fluxo) correspondentes aos elementos finitos e suas funções de forma. O pós-processamento opera sobre esses pares de pontos para esboçar as curvas de forma suave. Uma outra curva que pode ser obtida no pós-processamento é a  $\frac{dp(x)}{dx}$ , a qual é útil para analisar as perdas de carga ao longo da extensão do poço.

As curvas possibilitam o entendimento do comportamento do fluxo no interior do poço, além de auxiliar na prevenção de *breakthrough* de água ou gás precocemente, o que poderiam trazer aspectos adversos à exploração de petróleo em poços horizontais (Junior *et al.*, 2007). Além do mais, quando se busca a uniformização do fluxo ao longo do poço (Fernandes *et al.*, 2006) ou um estudo de influência de parâmetros sobre a produtividade (Vicente *et al.*, 2003), o uso dessas curvas se torna imprescindível.

# Capítulo 3

## Metodologia

A aplicação da rede neural artificial no presente trabalho baseia-se no grande potencial da rede tipo *MLP* em mapear funções a partir de um conjunto de padrões de treinamento a ela apresentado. Desta forma, foi desenvolvida uma metodologia para utilizar esse potencial para mapear um parâmetro de fluxo proveniente dos resultados do modelo tridimensional do acoplamento poço-reservatório. É gerado um conjunto numérico relativo a esse parâmetro e apresentado à rede neural como dados de treinamento. A *MLP*, após o processo de treinamento, fornece ao modelo simplificado unidimensional os valores de fluxo do reservatório, de forma que esse modelo gere resultados comparáveis com o modelo tridimensional. Essa etapa é desenvolvida a partir de um arranjo entre a *MLP* e o modelo simplificado. Para geração da *MLP*, foi desenvolvida uma biblioteca em C++, de forma a facilitar a aplicação junto ao modelo unidimensional.

Destacar-se-á primeiramente uma descrição geral da metodologia e das ferramentas computacionais utilizadas no trabalho. Posteriormente, descrever-se-á o modelo simplificado unidimensional do acoplamento poço-reservatório. Em seguida, serão descritos a metodologia para gerar o conjunto de dados e o treinamento da *MLP*, finalizando-se com a descrição do arranjo da *MLP* e o modelo unidimensional.

### 3.1 Descrição Geral

Toda a metodologia é baseada no desenvolvimento de arranjar as redes neurais artificiais com um modelo simplificado do acoplamento poço-reservatório. Para calibrar o modelo simplificado, utilizar-se-á um conjunto de dados contendo informações sobre um parâmetro de fluxo do modelo tridimensional como padrões de treinamento das redes neurais. Conforme será descrito em 3.3 e em 3.4, esse parâmetro de fluxo é definido como resistividade  $K(x)$ , variável ao longo do eixo do poço. A ideia fundamental é que essa resistividade  $K(x)$  mapeada pelas redes neurais possa ser utilizada

na resolução do modelo simplificado unidimensional e, assim, produzir resultados semelhantes aos obtidos através do modelo de elementos finitos. A Figura 3.1 ilustra o processo de arranjo entre as redes neurais e o modelo unidimensional, juntamente com a comparação entre os resultados do modelo de elementos finitos:

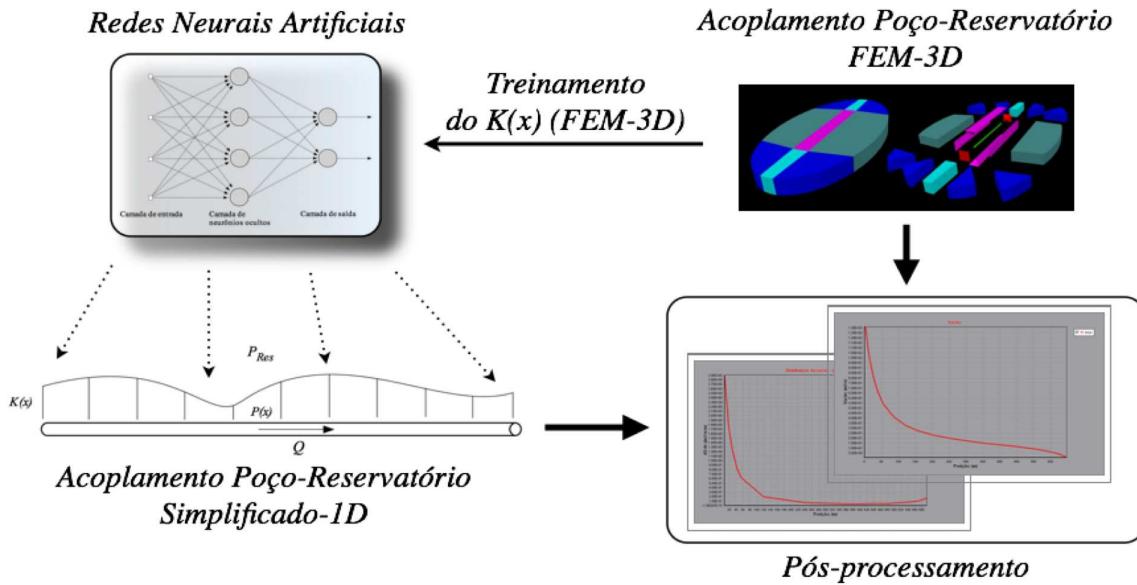


Figura 3.1: Arranjo das redes neurais e modelo 1D do acoplamento poço-resevatório

Conforme a Figura 3.1 apresenta, o modelo de elementos finitos fornece um conjunto de dados referentes à resistividade  $K(x)$  para treinamento das redes neurais; essas, após treinamento, são responsáveis por fornecer os valores de  $K(x)$  para o modelo simplificado unidimensional. Ao final, os resultados do modelo simplificado e do método dos elementos finitos deveriam ser, ao menos, próximos ou semelhantes. Em 3.6.2, são descritas as medidas de erros utilizadas para avaliar os resultados entre os dois modelos.

## 3.2 Ferramentas Computacionais Utilizadas

O trabalho foi desenvolvido e implementado num ambiente de programação em linguagem C++, o Xcode. Foi utilizado, além disso, um *software* de matemática simbólica como auxílio durante as implementações, validações e visualizações de gráficos bidimensionais e tridimensionais.

O modelo tridimensional do acoplamento poço-reservatório tem um *kernel* escrito em C++, possuindo entrada e saída de dados a partir de uma interface ou estrutura de dados ligada ao *kernel*.

Dessa forma, a geração e execução dos casos para composição do conjunto de treinamento, assim como a obtenção dos resultados, foram realizadas a partir do acesso e manipulação dessa estrutura de dados.

As *MLPs* foram definidas a partir de uma biblioteca desenvolvida para geração de redes neurais artificiais. Tal biblioteca, denominada *NeuralLib*, foi implementada na linguagem de programação C++. Possui estrutura típica de programação orientada a objetos. A *NeuralLib* foi concebida para permitir a criação e uso de redes neurais artificiais considerando vários tipos de neurônios, funções de ativação, arquitetura de redes e algoritmos de treinamento. Essa versatilidade deve-se à filosofia de programação orientada a objetos, a qual permite reutilização de códigos e facilidade de implementações em códigos existentes. Por ser fruto do presente trabalho, essa biblioteca não contempla ainda todas as tipologias pertinentes às redes neurais de forma geral. Assim, no momento, somente a arquitetura de rede do tipo *perceptron* de múltiplas camadas e o algoritmo *backpropagation* possuem implementações validadas. Implementações futuras, no entanto, poderão ser realizadas para desenvolvimento de tal biblioteca.

O trabalho utilizou o *kernel* do modelo tridimensional e a *NeuralLib* no mesmo ambiente de programação (*Xcode*), de forma a facilitar a interação entre as estruturas de dados e rotinas de cálculo. Detalhes da biblioteca desenvolvida estão em 4.1 e no Apêndice A.

### 3.3 Acoplamento Poço-Reservatório: Unidimensional

O modelo unidimensional do acoplamento poço-reservatório baseia-se nas formulações de fluxo descritas em 2.2.2 e 2.2.3. O modelo unidimensional modela, porém, o fluxo no reservatório de forma simplificada: representa-se o reservatório sobre o poço horizontal como uma espécie de resistividade  $K(x)$ , a qual gera uma resistência para o fluxo do reservatório em direção ao poço. A Figura 3.2 ilustra o modelo simplificado unidimensional.

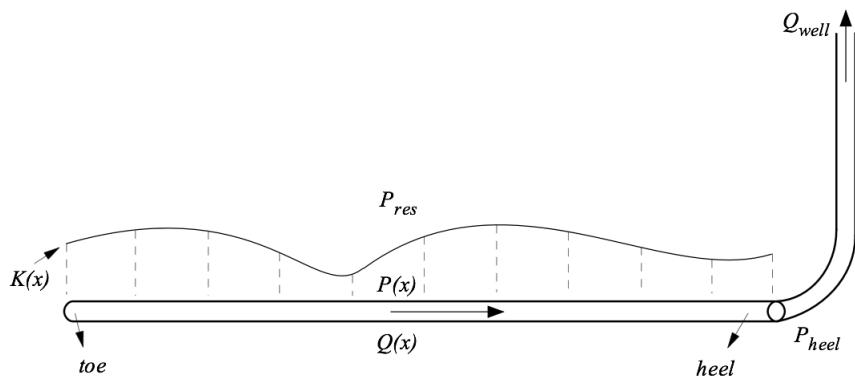


Figura 3.2: Modelo simplificado unidimensional do acoplamento poço-reservatório.

O equacionamento para este modelo pode ser desenvolvido analisando o fluxo no poço. Pela lei de conservação de massa, da Equação 2.26, tem-se:

$$\frac{dQ_{hw}(x)}{dx} + q_l(x) = 0. \quad (3.1)$$

O fluxo de óleo entrando no poço ( $q_l(x)$ ) é proporcional à diferença de pressão entre poço ( $p(x)$ ) e pressão no reservatório  $p_{res}$ . O fluxo pode ser equacionado como:

$$q_l(x) = K(x) \cdot (p(x) - p_{res}) \quad (3.2)$$

em que  $K(x)$  é a própria resistividade que representa o reservatório. Substituindo a Equação 3.2 em 3.1, tem-se:

$$\frac{dQ_{hw}(x)}{dx} + K(x) \cdot (p(x) - p_{res}) = 0 \quad (3.3)$$

ou

$$\frac{dQ_{hw}(x)}{dx} = -K(x) \cdot (p(x) - p_{res}). \quad (3.4)$$

Isolando  $K(x)$ , tem-se:

$$K(x) = -\frac{dQ_{hw}(x)}{dx} \cdot \frac{1}{(p(x) - p_{res})}. \quad (3.5)$$

O equacionamento para o escoamento no poço é o mesmo como apresentado em 2.2.3, ou seja:

$$\frac{dp(x)}{dx} = \frac{32f\rho Q_{hw}(x)^2}{\pi^2 D^5}. \quad (3.6)$$

Dessa forma, as equações diferenciais a serem resolvidas para o modelo unidimensional são compostas pelas Equações 3.4 e 3.6:

$$\frac{dQ_{hw}(x)}{dx} = -K(x) \cdot (p(x) - p_{res})$$

$$\frac{dp(x)}{dx} = \frac{32f\rho Q_{hw}(x)^2}{\pi^2 D^5}$$

em que  $f$  é o mesmo apresentado em 2.2.3.

As condições de contorno seguem as do modelo tridimensional:

- Pressão no calcanhar poço (*heel*) conhecida:  $P(x_{heel}) = P_{heel}$ , em que  $x_{heel}$  é a posição relativa ao calcanhar do poço;
- Vazão no dedão do poço (*toe*) igual a zero:  $Q_{hw}(x_{toe}) = Q_{toe} = 0$ , em que  $x_{toe}$  é a posição relativa ao dedão do poço;
- Pressão no reservatório  $p_{res}$  (*far field*) constante:  $p_{res} = \text{constante}$ .

O perfil da resistividade  $K(x)$  denota o comportamento de fluxo resultante da interação entre o reservatório e o poço produtor. A Figura 3.3 apresenta um exemplo da curva  $K(x)$  proveniente de um caso simulado a partir do modelo tridimensional. Ressalta-se que o eixo  $x$  representa o eixo do poço.

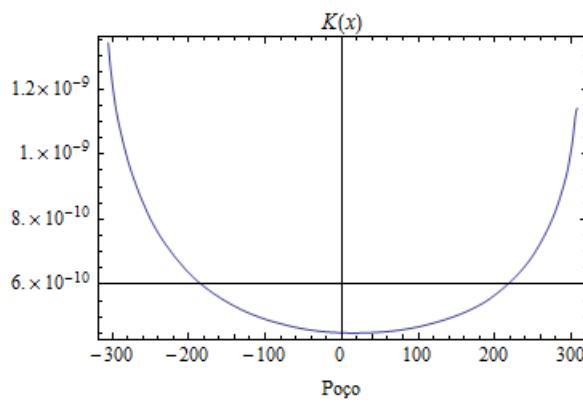


Figura 3.3: Exemplo da curva  $K(x)$ .

A Seção 3.4 descreverá uma interpretação e uma representação mais detalhada sobre a curva  $K(x)$ .

## 3.4 Análise da Resistividade $K(x)$

A Equação 3.5 indica que é possível obter o valor para resistividade  $K(x)$  a partir do valor de  $\frac{dQ_{hw}(x)}{dx}$  e  $p(x) - p_{res}$ . A resistividade  $K(x)$  pode ser interpretada como uma correção do fluxo no interior do poço devido à presença de fluxos esféricos nas extremidades do mesmo quando considerada a equação analítica para fluxo radial, aplicada em poços verticais. Será descrita a adimensionalização da resistividade partindo dessa interpretação. Na sequência, descrever-se-á a representação de  $K(x)$  por meio de polinômios.

### 3.4.1 Adimensionalização da Resistividade $K(x)$

Conforme apresentado em Rosa *et al.* (2006), a equação de fluxo radial permanente considerando fluido incompressível e perda de carga no interior do poço como desprezível é:

$$Q_w = \frac{2\pi k h (p_{res} - p_w)}{\mu \ln(r_{res}/r_w)} \quad (3.7)$$

em que  $Q_w$  é a vazão total do poço,  $k$  é a permeabilidade do meio poroso, tido como isotrópico,  $h$  é a altura do reservatório (igual ao comprimento do poço, considerando poço verticais),  $\mu$  é a viscosidade do fluido,  $r_{res}$  é o raio externo do reservatório,  $r_w$  é o raio do poço. Pode-se definir uma vazão por unidade de comprimento de poço que o reservatório pode fornecer, como segue:

$$q_{lw} = \frac{Q_w}{h} \quad (3.8)$$

a qual possui valor constante, já que é desprezada a perda de carga no interior do poço vertical. Considerando a lei de conservação de massa a partir de 3.1, tem-se:

$$\frac{dQ_w(x)}{dx} + q_{lw} = 0. \quad (3.9)$$

Utilizando-se de 3.8 e de 3.9, tem-se:

$$\frac{dQ_w(x)}{dx} = -\frac{Q_w}{h} = cte \quad (3.10)$$

o que leva à conclusão de que o valor de  $\frac{dQ_w(x)}{dx} = \frac{dQ_w}{dx}$  é constante para todo  $x$ , para um poço vertical. De 3.7, pode-se escrever:

$$\frac{Q_w}{h} = \frac{2\pi k (p_{res} - p_w)}{\mu \ln(r_{res}/r_w)}. \quad (3.11)$$

Substituindo  $\frac{Q_w}{h}$  em 3.11 a partir de 3.10, tem-se:

$$\frac{dQ_w}{dx} = -\frac{2\pi k (p_{res} - p_w)}{\mu \ln(r_{res}/r_w)} = \frac{2\pi k (p_w - p_{res})}{\mu \ln(r_{res}/r_w)}. \quad (3.12)$$

Rearranjando 3.12, tem-se:

$$\frac{dQ_w}{dx} \frac{1}{(p_w - p_{res})} = \frac{2\pi k}{\mu \ln(r_{res}/r_w)}. \quad (3.13)$$

Usando o conceito da resistividade pela 3.5, tem-se:

$$K_{vw} = -\frac{2\pi k}{\mu \ln(r_{res}/r_w)} \quad (3.14)$$

em que  $K_{vw}$  é a resistividade para poço vertical.

Embora a resistividade  $K_{vw}$  seja constante, ela não é adimensional. Pode-se definir uma resistividade adimensional,  $\bar{K}_{vw}$ , como se segue:

$$\bar{K}_{vw} = -\frac{\mu \ln(r_{res}/r_w)}{2\pi k} K_{vw} = 1. \quad (3.15)$$

O raciocínio de desenvolvimento de  $\bar{K}_{vw}$  pode ser utilizado para encontrar uma adimensionalização de  $K(x)$ .

Em poços horizontais, a geometria do poço e do reservatório fazem aparecer os fluxos esféricos nas extremidades do poço. Desta forma, o uso da Equação 3.7 em poços horizontais perde sentido e aplicabilidade. Porém, é possível ainda escrever uma equação com os mesmos princípios, porém com um fator de correção que contabilize os fluxos esféricos. Apenas com a motivação de se obter uma adimensionalização plausível da resistividade  $K(x)$  no caso de poços horizontais, pode-se escrever uma equação à semelhança da Equação 3.7:

$$Q_{hw} = \frac{2\pi k l_{hw} (p_{res} - p_{heel})}{\mu (B_{res}/H_{res})} \quad (3.16)$$

em que  $l_{hw}$  é o comprimento do poço,  $Q_{hw}$  é vazão total produzida pelo poço,  $B_{res}$  e  $H_{res}$  são largura e altura do reservatório, respectivamente. Ressalta-se que a Equação 3.16 apenas foi desenvolvida como motivação da adimensionalização que será descrita na sequência, não possuindo uma descrição fiel do problema físico.

Usando a Equação 3.16, pode ser obtida uma vazão por comprimento de poço, como se segue:

$$\frac{Q_{hw}}{l_{hw}} = \frac{2\pi k (p_{res} - p_{heel})}{\mu (B_{res}/H_{res})} = -\frac{2\pi k (p_{heel} - p_{res})}{\mu (B_{res}/H_{res})} \quad (3.17)$$

que é constante para todo o trecho. Pode-se relacionar essa vazão por comprimento com o  $\frac{dQ_{hw}(x)}{dx}$  real do poço horizontal:

$$\frac{dQ_{hw}(x)}{dx} = \frac{Q_{hw}}{l_{hw}} \alpha(x) = -\frac{2\pi k (p_{heel} - p_{res})}{\mu (B_{res}/H_{res})} \alpha(x) \quad (3.18)$$

em que  $\alpha(x)$  é um fator de correção requerido para contabilizar os fluxos esféricos. Definindo uma resistividade de forma análoga à Equação 3.14, tem-se:

$$K_{hw} = -\frac{2\pi k}{\mu (B_{res}/H_{res})}. \quad (3.19)$$

Da Equação 3.4, e utilizando essa resistividade na Equação 3.18, tem-se:

$$\alpha(x) = -\frac{K(x)(p(x) - p_{res})}{K_{hw}(p_{heel} - p_{res})}. \quad (3.20)$$

A Equação 3.18 pode ser escrita então como:

$$\frac{dQ_{hw}(x)}{dx} = -\frac{Q_w}{l_{hw}} \frac{K(x)(p(x) - p_{res})}{K_{hw}(p_{heel} - p_{res})}. \quad (3.21)$$

A partir da Equação 3.21, pode-se definir uma nova resistividade, como se segue:

$$\bar{K}(x) = \frac{K(x)}{K_{hw}}. \quad (3.22)$$

Essa nova resistividade é adimensional e pode ser interpretada como um fator de correção para contabilizar os fluxos esféricos sobre o  $Q_{hw}/l_{hw}$  proveniente da equação de escoamento radial. Para os objetivos do trabalho, optou-se por trabalhar com essa resistividade adimesional, principalmente para o treinamento das redes neurais. Comparando com a Equação 3.15, a resistividade  $\bar{K}(x)$ , além de não ser constante, não possui valor unitário como a  $\bar{K}_{vh}$ ; porém, ela tende a variar em torno do valor unitário por conta dos fluxos esféricos.

### 3.4.2 Representação de $K(x)$

Além do processo de adimensionalizar, foi necessário representar a resistividade com poucos parâmetros, já que não há uma expressão analítica que a defina, principalmente por se tratar de um pós-processamento dos resultados do método dos elementos finitos. Desta forma, optou-se em representá-la através de uma forma polinomial. Devido às características inerentes aos polinômios ortogonais, tais como ortogonalidade em um intervalo, raízes distintas nesse intervalo etc. (Cunha, 2009; Burden e Faires, 2008), optou-se por utilizar esse tipo de função. Dentre outros, escolheu-se os polinômios de *Legendre*, que são ortogonais em  $[-1, 1]$  com relação à função peso  $\omega(x) \equiv 1$  (Burden e Faires (2008)).

A representação da resistividade  $K(x)$  dada pela Equação 3.5, isto é, dimensional, pode gerar certos problemas numéricos, devido à geração de coeficientes polinomiais com valores relativamente pequenos devido a alta ordem dos valores de pressão. Isso justifica a adimensionalização descrita em 3.4.1. Além do mais, por ser uma função descrita ao longo do eixo do poço, a adimensionalização do eixo  $x$  também torna-se justificável. Assim, para estar coerente com o intervalo onde os polinômios de *Legendre* são ortogonais, optou-se escalonar o eixo  $x$ , definido originalmente entre  $[-\frac{l_{hw}}{2}, \frac{l_{hw}}{2}]$ , para  $\bar{x} \in [-1, 1]$ . Dessa forma, a resistividade adimensional  $\bar{K}(\bar{x})$  pode ser aproximada por uma função polinomial como:

$$\bar{K}(\bar{x}) \cong \bar{K}_p(\bar{x}) = a_n L_n(\bar{x}) + a_{n-1} L_{n-1}(\bar{x}) + \dots + a_1 L_1(\bar{x}) + a_0 \quad (3.23)$$

em que  $L_n(\bar{x})$  é o  $n$ -ésimo polinômio de *Legendre*,  $a_n$  são coeficientes multiplicadores e o subíndice  $p$  indica que é função polinomial.

Os resultados provenientes do modelo de elementos finitos são expressos por pares de pontos, os quais relacionam coordenadas geométricas e valores de uma dada solução, conforme descrito na Subseção 2.2.4. Dessa forma, para representar de forma polinomial, foi necessário ajustar os pares de pontos referentes à resistividade  $K(x)$  através de regressão linear. Foi utilizado o método dos quadrados mínimos sobre o conjunto de resultados para calcular os coeficientes  $a_n$ . Testes considerando o valor numérico de resíduo proveniente da regressão linear revelaram que um polinômio de grau 6 adequa-se suficientemente a tal ajuste. A metodologia para realizar a regressão linear está descrita em Cunha (2009). Abaixo segue a sequência de passos para obtenção de  $\bar{K}_p(\bar{x})$ , para cada caso do conjunto de treinamento (vide Subseção 3.5.1):

1. Cálculo de  $K(x)$ , a partir dos pares de pontos:

$$K(x_i) = \left\{ -\frac{dQ_{hw}(x_i)}{dx} \cdot \frac{1}{(p(x_i) - p_{res})} \right\}_{FEM-3D}$$

em que  $x_i$  são os pontos no eixo do poço e o subíndice  $FEM - 3D$  indica que os argumentos são provenientes dos resultados do modelo de elementos finitos.

2. Adimensionalização de  $K(x_i)$  e cálculo de  $\bar{K}(x_i)$ :

$$\bar{K}(x_i) = \frac{K(x_i)}{K_{hw}}$$

em que  $K_{hw}$  é dado pela Equação 3.19.

3. Reescalonamento dos  $x_i$ :

$$\bar{x}_i = \frac{2 \cdot x_i}{l_{wh}}$$

sendo  $l_{wh}$  o comprimento total do poço horizontal.

4. Composição dos novos pares de pontos:

$$(\bar{x}_i, \bar{K}(\bar{x}_i)), i = 1, \dots, n_p$$

em que  $n_p$  é o número de pontos os quais o modelo de elementos finitos gera durante o pós-processamento.

5. Regressão linear - cálculo de  $\bar{K}_p(\bar{x})$ :

Encontrar os coeficientes  $a_n$ ,  $n = 0,..,6$ , tal que minimize a função objetivo:

$$E_7(a_0, \dots, a_6) = \sum_{i=1}^{n_p} [\bar{K}(\bar{x}_i) - \bar{K}_p(\bar{x}_i)]^2 \quad (3.24)$$

em que  $\bar{K}_p(\bar{x}_i) = a_6 L_6(\bar{x}_i) + \dots + a_1 L_1(\bar{x}_i) + a_0$ .

O uso da regressão linear permite utilizar a resistividade  $\bar{K}_P(\bar{x})$  no lugar de  $K(x)$  sem maiores perdas de precisão, trazendo a vantagem de ser descrita através de um número pequeno de parâmetros,  $a_n$ ,  $n = 0,..,6$ . Esses parâmetros, como será descrito na Subseção 3.5.1, são os valores esperados do conjunto de treinamento da *MLP*.

## 3.5 Aprendizagem da Rede Neural Artificial

A representação da resistividade  $K(x)$  na forma polinomial ( $\bar{K}_p(\bar{x})$ ), conforme descrito em 3.4.2, possibilita gerar conjunto de dados passíveis de serem usados no treinamento de uma rede neural tipo *MLP*. A geração deste conjunto de dados e o processo de aprendizagem da rede neural são descritos a seguir.

### 3.5.1 Conjunto de Treinamento

A cada cálculo que se faça usando o modelo de elementos finitos tridimensional, um polinômio de grau 6 é encontrado de forma a descrever a resistividade  $K(x)$  para tal simulação. Ou seja, em cada simulação, existem 7 parâmetros  $a_n$ ,  $n = 0,..,6$ , que descrevem a resistividade adimensionalizada  $\bar{K}_p(\bar{x})$ . Esse conjunto de parâmetros para uma dada simulação (ou caso)  $k$  será denominado como  $\vec{a}(k)$ , isto é, vetor de saída esperado do  $k$ -ésimo caso do conjunto de treinamento, à semelhança do vetor  $\vec{d}(k)$  descrito em 2.1.4; portanto:  $\vec{a}(k) \equiv \vec{d}(k)$ .

Conforme descrito em 2.2.1, o modelo tridimensional do acomplamento poço-reservatório apresenta uma gama de dados de entrada ou variáveis *inputs* necessárias para definição do problema físico a ser simulado. Além do mais, cada *input* ou variável pode assumir qualquer valor dentro de um intervalo relativamente grande, de acordo com a natureza física da variável. Por exemplo, a variável “permeabilidade” (horizontal e vertical) terá seu valor extremamente dependente do tipo de rocha pela qual o reservatório é constituído. Assim, selecionou-se algumas variáveis de *input* para formar os sinais de entrada da rede neural, com intuito de observar a capacidade e limitação da *MLP* quando do treinamento e generalização de valores. Para cada variável selecionada, foi estipulado um intervalo de variação a partir de valores usuais de campo, de forma a respeitar o modelo

físico. Algumas considerações adicionais com relação ao modelo do reservatório e do poço foram assumidas para o presente trabalho:

- Área de drenagem retangular;
- Completação de poço tipo aberto;
- Fator de dano (*skin*) nulo;
- Comprimento do poço (*m*): 400,0;
- Pressão no reservatório - *far field* (*kgf/cm<sup>2</sup>*): 225,0;

As variáveis de *input* selecionadas e os intervalos de variação utilizados para geração dos casos são:

- Raio de drenagem (*m*), intervalo: [400,0; 1600,0];
- Altura do reservatório (*m*), intervalo: [15,0; 60,0];
- Afastamento vertical (*m*), intervalo: [5,0; 55,0];
- Pressão no poço - *heel* (*kgf/cm<sup>2</sup>*), intervalo: [117,0; 225,0];
- Raio do poço (m), intervalo: [0,05; 0,10];
- Viscosidade do óleo (*Pa · s*), intervalo: [0,001; 0,007];
- Massa específica do óleo (*kg/m<sup>3</sup>*), intervalo: [750,0; 950,0].

As permeabilidades horizontal e vertical do reservatório foram consideradas como constantes, sendo adotado um valor típico. Assim, essas 7 variáveis formam os sinais de entrada da rede neural, os quais serão definidos como  $\vec{\chi}(k)$ , isto é, vetor de sinal de entrada para o *k*-ésimo caso do conjunto de treinamento.

Ao todo, foram gerados 100 casos a partir do modelo tridimensional variando de forma aleatória as variáveis de *input*. Esses 100 casos formam o conjunto de dados para os quais as redes neurais foram treinadas, sendo definido como:

$$A = \{(\vec{\chi}(k), \vec{d}(k)) : \vec{d}(k) = f_{FEM-3D}(\vec{\chi}(k)), k = 1, \dots, n_t, n_t \in \mathbb{N}\} \quad (3.25)$$

em que  $f_{FEM-3D}(\vec{\chi})$  representa o modelo tridimensional, e  $n_t = 100$ . Os casos foram numerados de 1 a 100 para facilitar a análise individual.

Esse conjunto de dados foi dividido em três subconjuntos, a saber:

- Subconjunto de treinamento. Esse é o conjunto de casos utilizados para ajustar os pesos sinápticos da rede neural através do algoritmo de treinamento. O subconjunto é formado por 50 casos escolhidos de forma aleatória, representando 50,0% do conjunto  $A$ . Define-se esse subconjunto como:

$$A_T \subset A | A_T = \{(\vec{\chi}(k), \vec{a}(k)) : \vec{a}(k) = f_{FEM-3D}(\vec{\chi}(k)), k = 1, \dots, 50\}. \quad (3.26)$$

- Subconjunto de validação. Esse é o conjunto de casos utilizados para analisar o desempenho do processo de treinamento da rede neural. O subconjunto é utilizado de forma a evitar o chamado "supertreinamento" ou *overfitting*, isto é, quando a rede neural perde a capacidade de extrair as caracterísitcas gerais do subconjunto de treinamento e começa a produzir generalizações ruins. Assim, o uso desse subconjunto permite determinar o número de iterações máximo no processo de treinamento. O termo *overfitting* será empregado no decorrer do texto. O subconjunto é composto por 25 casos, compondo 25,0% do conjunto  $A$ . É definido como:

$$A_v \subset A | A_v = \{(\vec{\chi}(k), \vec{a}(k)) : \vec{a}(k) = f_{FEM-3D}(\vec{\chi}(k)), k = 51, \dots, 75\}. \quad (3.27)$$

- Subconjunto de teste. Esse subconjunto não participa diretamente do processo de treinamento, porém tem a finalidade de avaliar o estado da rede neural após treinamento com relação às generalizações. O subconjunto é composto por 25 casos, ou seja, 25,0% do conjunto  $A$ . Define-se esse subconjunto como:

$$A_{te} \subset A | A_{te} = \{(\vec{\chi}(k), \vec{a}(k)) : \vec{a}(k) = f_{FEM-3D}(\vec{\chi}(k)), k = 76, \dots, 100\}. \quad (3.28)$$

Assim,  $A_T \cup A_v \cup A_{te} = A$ .

Cabe ressaltar que, após a geração dos 100 casos, foi realizada uma análise entre os casos dos subconjuntos  $A_v$  e  $A_{te}$  e os do subconjunto de treinamento  $A_T$ . O objetivo dessa análise foi de verificar se os casos dos subconjuntos  $A_v$  e  $A_{te}$  estavam no interior do hipercubo gerado pelos elementos do subconjunto  $A_T$ . Essa ressalva foi feita para garantir que todos os casos que não estivessem no subconjunto de treinamento ( $A_T$ ) fossem casos de interpolação e não de extrapolação. A análise adotada baseou-se em verificar individualmente se cada variável de *input* e *output* dos casos de  $A_v$  e  $A_{te}$  estavam entre os valores extremos das mesmas variáveis nos casos de  $A_T$ . Um procedimento mais aprimorado pode ser encontrado em Bazaraa *et al.* (1993), no qual são descritos algoritmos para análise de conjuntos (*convex sets* e *convex hulls*). Embora se tenha procurado evitar que a *MLP* extrapolasse algum caso gerado, sabe-se que as redes neurais são, idealmente, capazes de mapear valores de funções mesmo estando além do intervalo de treinamento. No presente trabalho, contudo, preferiu-se utilizar as redes neurais apenas como funções interpoladoras.

A análise e critério de erros do processo de treinamento serão discutidos na Subseção 3.5.3. A estrutura de rede adotada é descrita na sequência.

### 3.5.2 Arquitetura de Rede Neural Utilizada

A arquitetura de rede utilizada no presente trabalho é o *perceptron* de múltiplas camadas, ou *MLP*, descrito com detalhes em 2.1.3. A estrutura adotada possui duas camadas de processamento, sendo uma delas oculta. Embora o objetivo do trabalho não seja encontrar a arquitetura ideal para o problema, optou-se por gerar três *MLPs*, variando-se a quantidade de neurônios na camada oculta, com intuito de comparar os resultados de cada uma. De acordo com o teorema da aproximação universal, descrito em Haykin (2001), uma única camada oculta é suficiente para um perceptron de múltiplas camadas computar uma aproximação  $\varepsilon$  uniforme para um dado conjunto de treinamento representado pelo conjunto de entradas e a saídas-alvo. Porém, o teorema não diz que a única camada oculta é ótima no sentido do tempo de aprendizagem, facilidade de implementação ou generalização (Haykin, 2001), além de não prever a quantidade de neurônios necessários para a aproximação  $\varepsilon$  adotada. Ou seja, trata-se de um teorema existencial, o qual justifica matematicamente a aproximação de uma função contínua arbitrária por uma rede *perceptron* de uma camada oculta. Dessa forma, a comparação entre as três arquiteturas adotadas ilustrará as capacidades e limitações de cada uma. Optou-se por gerar as três redes com 5, 10 e 15 neurônios na camada oculta, sendo denominadas como *MLP-5*, *MLP-10* e *MLP-15*, respectivamente. A camada de entrada possui 7 neurônios e a de saída, 7, de acordo com as variáveis de *input* e *output*, respectivamente.

Escolheu-se como função de ativação dos neurônios a função sigmoidal (tangente hiperbólica), que é uma função anti-simétrica. Para os parâmetros  $a$  e  $b$  da função, conforme apresentado em 2.8, adotou-se os seguintes valores:

$$a = 1,7159$$

e

$$b = \frac{2}{3}.$$

De acordo com Haykin (2001), esses valores em especial produzem algumas propriedades úteis no que se diz ao comportamento da função tangente hiperbólica. Pode-se listar essas propriedades como:

- $\varphi(1) = 1$  e  $\varphi(-1) = -1$ ;
- A inclinação da função de ativação fica próxima da unidade na origem, ou seja,

$$\varphi'(0) = ab = 1,7159 \cdot 2/3 = 1,1424.$$

- A derivada segunda de  $\varphi(v)$  atinge seu valor máximo em  $v = 1$ .

### 3.5.3 Processo de Aprendizagem

O tipo de treinamento utilizado é o bem conhecido algoritmo retropropagação ou *backpropagation*, conforme descrito em 2.1.4. Para sua utilização, alguns parâmetros inerentes do algoritmo precisaram ser definidos. Algumas heurísticas também foram adotadas com o intuito de aumentar o desempenho do processo de aprendizagem. As heurísticas são descritas na sequência:

- Taxa de treinamento:  $\eta = 0,05$ . Foi desenvolvida uma função que altera esse valor com o número de iterações de treinamento. A ideia é que, à medida que o número de alterações avança, a taxa de crescimento assume valores cada vez menores, implementando assim uma heurística apresentada em Haykin (2001);
- Número máximo de iterações: 10.000. O número máximo de iterações foi determinado para evitar o *overfitting* da rede. Porém, o número de iterações para cada arquitetura de rede foi determinado através do processo de validação juntamente com os ajustes dos pesos sinápticos;
- Critério de parada: erro do subconjunto de treinamento, erro subconjunto de validação ou número máximo de iterações. O erro adotado para o subconjunto de treinamento foi a energia média do erro  $\xi_{med}(n)$ , conforme definida por 2.12;
- Constante de momento:  $\alpha = 10^{-7}$ . Conforme apresentado na Seção 2.1.4, optou-se em utilizar uma ponderação do valor do peso sináptico da iteração anterior, compondo assim o ajuste dado pela regra delta generalizada;
- Modo sequencial de aprendizagem. Os pesos sinápticos são atualizados a cada passo de propagação do algoritmo *backpropagation*, e não no final de cada época;
- Normalização dos padrões de entrada e saída. Todos os valores de entrada e saída de todos os casos foram escalonados entre  $[-1, 1]$ . Além disso, antes do escalonamento, foi feita uma remoção da média de cada parâmetro, tanto do vetor de entrada quanto do vetor de saída;
- Inicialização dos pesos e *bias* (polarizador). Todos os pesos sinápticos foram inicializados com números randômicos, mantendo o cuidado de não serem grandes suficientes para saturarem os neurônios, nem pequenos a ponto de atrasarem o aprendizado. Os *biases* foram inicializados com valores nulos.

O treinamento das três arquiteturas de rede foi feito utilizando os dois subconjuntos de dados: o de treinamento e o de validação. O primeiro subconjunto é o que atualiza os pesos sinápticos a partir do algoritmo de retropropagação ou *backpropagation*. O segundo é utilizado para avaliar a generalização da rede, de forma a evitar o *overfitting*. Embora o objetivo do trabalho seja avaliar

os resultados gerados pelo arranjo das redes neurais com o modelo unidimensional, o processo de treinamento das redes somente se deu pelos dados referentes à curva  $\bar{K}_p(\bar{x})$ . No entanto, o bom treinamento da rede é imprescindível para que os erros nos resultados finais possam ser minimizados. Isso será explicado na próxima seção e observado no Capítulo 4.

## 3.6 MLP e o Modelo Unidimensional

O processo de arranjo da rede neural com o modelo unidimensional baseia-se em utilizar a *MLP* treinada como a função que fornece o valor da resistividade  $\bar{K}_p(\bar{x})$  no processo de resolução das Equações 3.4 e 3.6.

Procurou-se resolver as equações do modelo unidimensional utilizando método numérico de solução de equações diferenciais. Segundo Cunha (2009), os métodos de *Runge-Kutta* são os mais usados dentre aqueles apropriados para os problemas de valor inicial. Existem variações do método, mas de acordo com Cunha (2009), o método de *Runge-Kutta* de quarta ordem é mais usado por ser uma combinação de simplicidade, alta precisão e economia. Desta forma optou-se em usar o método *Runge-Kutta* de quarta ordem.

### 3.6.1 Resolução do Modelo Unidimensional

As Equações 3.4 e 3.6, juntamente com as condições de contorno, não podem ser resolvidas usando o método de *Runge-Kutta* de forma direta, já que este método aplica-se a problemas de valor inicial. Sendo assim, foi preciso "transformar" esse sistema de equações com condições de contorno num sistema equivalente com condições iniciais. O dedão (*toe*) do poço foi escolhido para ser o ponto sobre o qual as condições iniciais são definidas. De acordo com as condições definidas em 3.3, a vazão no *toe* ( $Q_{hw}(x_{toe}) = Q_{toe} = 0$ ) será aplicada sem qualquer alteração. Para que a variável  $p(x)$  tivesse uma condição inicial, optou-se em usar um diferencial  $\Delta p$  para calcular o valor inicial da pressão no *toe*. Isto é:

$$p(x_{toe}) = p_{toe} = p_{res} - \Delta p. \quad (3.29)$$

Inicialmente o valor  $\Delta p$  não é conhecido; assim, é adotado um valor tal que  $p_{toe}$  seja maior do que  $p_{heel}$ . Após a divisão do domínio do poço em intervalos iguais, aplica-se o método de *Runge-Kutta* de quarta ordem, conforme descrito em Cunha (2009) e em Press *et al.* (2007). Por serem equações acopladas, o método foi aplicado para as duas equações num esquema iterativo: calculou-se primeiramente o fator  $m_0$  para a vazão  $Q_{hw}$  e, em seguida, o fator  $m_0$  para a pressão  $p$ ; repetiu-se esse procedimento para cálculo dos  $m_i$ ,  $i = 1, 2, 3$  até o cálculo de  $Q_{hw}^{k+1}$  e de  $p^{k+1}$ , sendo  $k$  o ponto do intervalo no domínio do poço. Ao final do procedimento, existirão  $k + 1$  pontos de valores para  $Q_{hw}$  e para  $p$ . O procedimento numérico somente é finalizado quando o valor  $p^{k+1}$ ,

correspondente ao valor da pressão no *heel*, se aproximar, a menos de uma tolerância, do valor  $p_{heel}$ . Essa tolerância foi definida em  $10^{-5}$ . Caso a diferença entre tais valores esteja além da tolerância, o  $\Delta p$  é corrigido e aplica-se novamente o método de *Runge-Kutta*, compondo assim um procedimento iterativo.

Ressalta-se que o valor da resistividade que aparece na Equação 3.4 é fornecido pela *MLP* treinada. Embora a rede neural forneça o valor adimensionalizado da resistividade ( $\bar{K}_p$ ), esse é convertido facilmente no valor usual ( $K_p$ ) a partir da Equação 3.22, utilizando-se para isso os parâmetros que compõe a resistividade  $K_{hw}$ . Esse procedimento é realizado em cada passo do *Runge-Kutta*.

A finalização deste processo iterativo resultará num conjunto de  $k + 1$  pontos correspondentes aos valores de  $Q_{hw}(x)$ ,  $p(x)$ ,  $\frac{dQ_{hw}(x)}{dx}$  e  $\frac{dp(x)}{dx}$ , além do valor da vazão total  $Q_{hw}$ . Esses conjuntos de pontos poderão ser esboçados em gráficos e comparados com as curvas originais provenientes do modelo de elementos finitos, assim como o valor da vazão do poço. A análise desses resultados a partir de uma norma de erro será descrito na Subseção 3.6.2.

### 3.6.2 Análise dos Resultados: Medida de Erros

Os resultados obtidos com o arranjo *MLP*-Modelo1D devem ser comparados com os resultados originais do modelo de elementos finitos perante algum critério ou medida de erro. Isso permitirá avaliar a utilização do arranjo em substituição ao modelo tridimensional, motivação e objetivo do trabalho. Assim, adotou-se como medida de erro a norma  $L^2$  para avaliar as curvas  $Q_{hw}(x)$ ,  $p(x)$ ,  $\frac{dQ_{hw}(x)}{dx}$  e  $\frac{dp(x)}{dx}$ , que é uma norma muito comum em simulações de elementos finitos (Becker *et al.*, 1981). Para a vazão total do poço,  $Q_{hw}$ , optou-se em usar a norma  $L^1$ , porém de forma alterada, a fim de que possa ser expressa a partir de um percentual.

A norma  $L^2$  para uma função  $f(x)$  contínua para um intervalo  $[a, b]$  é definida, de acordo com Kreyszig (1978):

$$\|f(x)\|_{L^2} = \sqrt{\int_a^b [f(x)]^2 dx}. \quad (3.30)$$

Assim, para avaliar uma medida de erro entre as curvas, usou-se:

$$E_f = \sqrt{\int_0^{l_{hw}} [f(x) - \bar{f}(x)]^2 dx} \quad (3.31)$$

em que  $f(x)$  representa as curvas do modelo de elementos finitos,  $\bar{f}(x)$  as curvas do arranjo *MLP*-modelo1D e  $l_{hw}$  é o comprimento do poço horizontal. Dessa forma, contabiliza-se os erros para as

quatro curvas em questão:  $E_Q$ ,  $E_{dQdx}$ ,  $E_p$  e  $E_{dpdx}$ .

Para a vazão total do poço, usou-se a norma  $L^1$ , escrita como:

$$E_{Q_{hw}}^{L^1} = |Q_{hw} - \bar{Q}_{hw}| \quad (3.32)$$

em que  $Q_{hw}$  é a vazão original do modelo de elementos finitos e  $\bar{Q}_{hw}$  do arranjo *MLP-modelo1D*. Porém, por se tratar de um número representativo em termos da análise de produtividade do poço, optou-se em utilizar uma norma  $L^1$  alterada. Dessa forma, tem-se uma medida de erro relativo, dado como:

$$E_{Q_{hw}} = \frac{|Q_{hw} - \bar{Q}_{hw}|}{Q_{hw}}. \quad (3.33)$$

A Equação 3.33 também pode ser expressa de forma percentual.

As normas de erros representadas pelas Equações 3.31 e 3.33 foram aplicadas para todos os casos dos três subconjuntos (treinamento, validação e teste), para cada uma das três arquiteturas. Para analisar o comportamento de cada subconjunto, optou-se em fazer a média aritmética das normas dos erros para cada um. Assim, é possível verificar qual arquitetura produz melhores resultados (menores erros), principalmente para os casos do subconjunto de teste  $A_{te}$ , pelo qual estaria avaliando a qualidade de generalização das *MLPs*. Optou-se em utilizar, juntamente com a média, o desvio padrão amostral  $\sigma$  (Bussab e Morettin, 1997) como forma de avaliar a variabilidade dos valores dos erros de cada subconjunto, em cada uma das arquiteturas.



# Capítulo 4

## Resultados

Os resultados foram obtidos de acordo com a metodologia apresentada no Capítulo 3. Esses se dividem em três partes: a primeira apresenta a biblioteca escrita em C++ para desenvolvimento de redes neurais artificiais, *NeuralLib*; a segunda parte apresenta os resultados de treinamento da rede neural, juntamente com algumas análises acerca do processo de aprendizagem; a terceira e última parte apresenta a comparação dos resultados obtidos com o modelo unidimensional frente aos padrões do modelo tridimensional.

### 4.1 *NeuralLib*

Um dos objetivos do trabalho era o desenvolvimento de uma biblioteca que permitisse compor redes neurais artificiais a fim de serem facilmente aplicadas na mecânica computacional e simulação numérica. O desenvolvimento em C++ foi escolhido devido às características desejáveis de tal linguagem em simulação e análise numérica, tais como orientação a objetos, herança e polimorfismo, concepção em *templates*, entre outros (Lippman e Lajoie, 1998). O uso de *templates* nesse caso foi necessário pois, assim, é possível estender a biblioteca para diversos tipos de neurônios, funções de ativação, arquiteturas de rede e mesmo padrões de aprendizagem, compondo assim de fato a caracterização de uma biblioteca. Além do mais, a concepção do código permite facilidade para suporte e implementação de funcionalidades diversas, tais como análise da evolução do erro de treinamento, análise de problemas com saturação dos neurônios etc. Os detalhes de como as classes *template* foram desenvolvidas e arranjadas estão no Apêndice A.

Alguns testes foram realizados para analisar e validar o código implementado. Além do mais, alguns testes serviram como base para entender o funcionamento e comportamento das redes neurais artificiais. Abaixo, seguem algumas funções utilizadas como forma de verificar a potencialidade de generalização das *MLPs*. As equações originais e os gráficos por estas produzidas são

apresentados. Para treinamento das *MLPs*, foram gerados somente conjuntos de treinamento.

- Função *Peaks*:

Essa função é definida pela função 4.1:

$$f(x, y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5}-x^3-y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}. \quad (4.1)$$

O gráfico dessa função é apresentado pela Figura 4.1 :

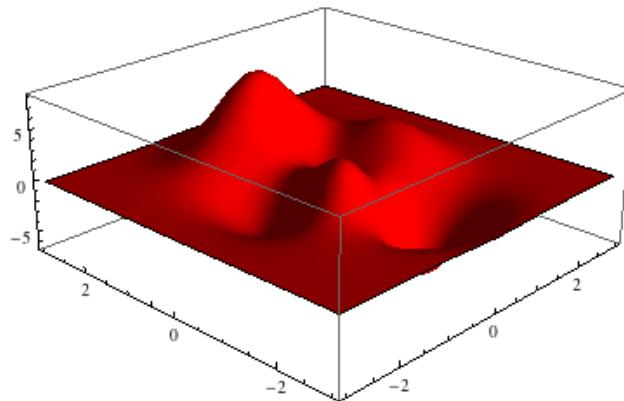


Figura 4.1: Função *Peaks*.

A Figura 4.2 abaixo apresenta o gráfico da função original (em vermelho) e o gráfico produzido por uma *MLP* gerada pela *NeuralLib* (em verde):

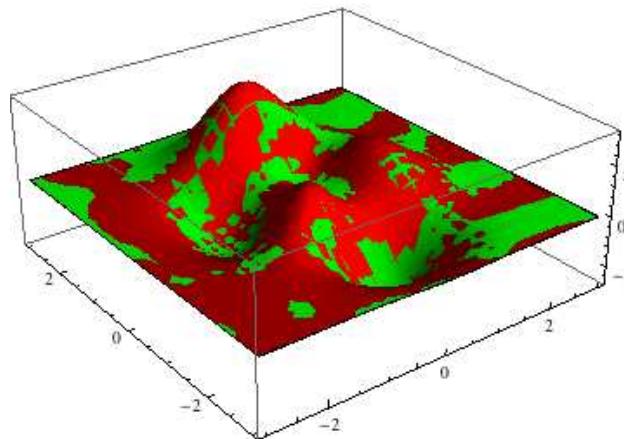


Figura 4.2: Comparação entre os gráfico original e o gerado pela *MLP*.

Observa-se que os dois gráficos confundem-se. Para ter uma medida da aproximação, usou-se a norma  $L^2$  da diferença  $E_{peak}$  entre as duas superfícies dividida pela norma  $L^2$  da função original,  $\|f_{peak}\|_{L^2}$ , sendo que  $E_{peak}$  é obtida de forma semelhante à 3.31. Assim, obteve-se:

$$\bar{E}_{peak} = \frac{E_{peak}}{\|f_{peak}\|_{L^2}} = \frac{0,2605}{11,6448} = 0,0224.$$

O valor da norma  $\bar{E}_{peak}$  indica boa aproximação da função original.

- Função descontínua:

A função descontínua utilizada apresenta a seguinte expressão, definida pela função 4.2:

$$f(x, y) = \begin{cases} x^2 + y^2 - 25 & sex < -7 \\ -2 \sin x - \frac{xy^2}{10} + 15 & sex < -3 \\ 0,5x^2 + 20 + |y| & sex < 0 \\ 0,3\sqrt{x} + 25 + |y| & sex \geq 0 \end{cases} \quad (4.2)$$

com  $-7,5 \leq x \leq 3,0$  e  $-3,0 \leq y \leq 3,0$ .

O gráfico dessa função é apresentado pela Figura 4.3:

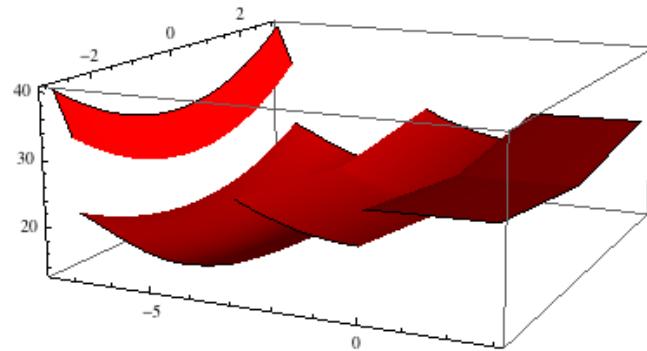


Figura 4.3: Função descontínua.

A Figura 4.4 abaixo apresenta o gráfico da função original (em vermelho) e o gráfico produzido por uma *MLP* gerada pela *NeuralLib* (em verde):

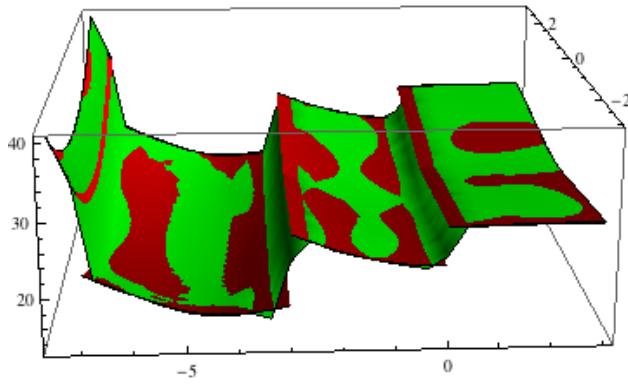


Figura 4.4: Comparação entre os gráfico original e o gerado pela *MLP*.

Observa-se que, embora a função original seja descontínua, a *MLP* conseguiu reproduzir os trechos contínuos com boa acurácia. As descontinuidades foram mapeadas através de trechos com gradiente alto, o que é plausível, já que as funções de ativação dos neurônios são todas contínuas. Para analisar a aproximação, calculou-se a norma  $L^2$  da diferença  $E_{nSmooth}$  entre as duas superfícies dividida pela norma  $L^2$  da função original,  $\|f_{nSmooth}\|_{L^2}$ , calculada em cada trecho onde ela é contínua. Assim, obteve-se:

$$\bar{E}_{nSmooth} = \frac{E_{nSmooth}}{\|f_{nSmooth}\|_{L^2}} = \frac{8,1925}{189,6232} = 0,0432.$$

O valor da norma  $\bar{E}_{nSmooth}$  indica que, mesmo sendo uma aproximação contínua de uma função descontínua, o valor do erro em  $L^2$  para esse caso não ultrapassa 5% da norma da função original.

- Função *Rastrigin*:

Essa função é definida pela função 4.3:

$$f(x, y) = 20 + (x^2 + y^2) - 10(\cos 2\pi x + \cos 2\pi y) \quad (4.3)$$

com  $-5,0 \leq x, y \leq 5,0$ .

O gráfico dessa função é apresentado pela Figura 4.5:

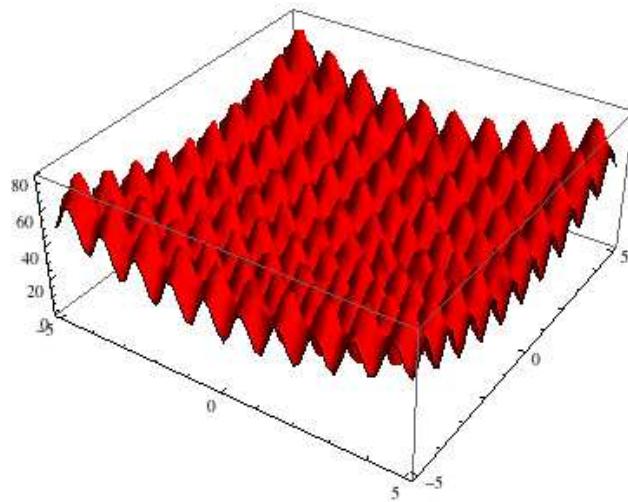


Figura 4.5: Função *Rastrigin*.

A Figura 4.6 abaixo apresenta o gráfico da função original (em vermelho) e o gráfico produzido por uma *MLP* gerada pela *NeuralLib* (em verde):

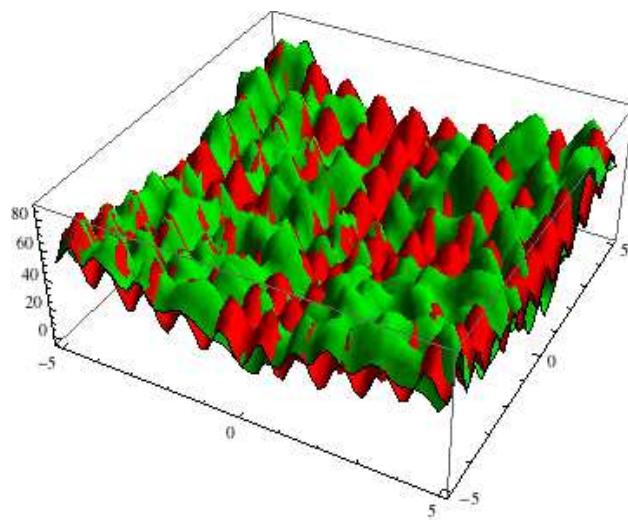


Figura 4.6: Comparaçāo entre os gráfico original e o gerado pela *MLP*.

Observa-se a *MLP* consegue mapear globalmente a função original, isto é, preserva os limites máximos e mínimos da função *Rastrigin*. Nem todas as oscilações são captadas pela *MLP*, porém a generalização global apresenta bom comportamento. Ressalta-se que nesse caso as funções de ativação dos neurônios foram definidas através de funções senoidais. Isso foi feito com objetivo de aumentar a percepção da *MLP* frente às oscilações da função original. O valor da norma  $L^2$  da diferença entre as superfícies normalizada pela norma da superfície original é:

$$\bar{E}_{Rast} = \frac{E_{Rast}}{\|f_{Rast}\|_{L^2}} = \frac{131,0932}{396,4206} = 0,3307.$$

Percebe-se que o valor da norma da diferença representa cerca de 30% da norma da função original. Embora o erro seja razoavelmente grande, o mapeamento realizado pelas redes neurais capta globalmente a topologia da função *Rastrigin*.

## 4.2 Aprendizagem da Rede Neural

Essa seção descreve os resultados do treinamento das *MLPs*. Para cada uma das arquiteturas definidas foram esboçados gráficos das curvas de evolução das energias médias do erro, tanto para o subconjunto  $A_T$  quanto para o  $A_v$ . Foram também compostas tabelas contendo as energias médias do erro ao final do treinamento para os três subconjuntos,  $A_T$ ,  $A_v$  e  $A_{te}$ . Juntamente com as energias médias dos subconjuntos, foram acrescentados valores dos desvios padrão. O uso do desvio padrão amostral  $\sigma$  tem o intuito de avaliar a dispersão do valor da energia total do erro  $\xi(k)$  entre os  $k$ -ésimos casos presentes nos três subconjuntos. Além do mais, servirá como um indicativo na comparação entre as três arquiteturas utilizadas. Para comparar os resultados de treinamento entre as arquiteturas foi composto um gráfico com as energias totais de cada caso do conjunto  $A$ .

### 4.2.1 Arquitetura 1: MLP-5

As Figuras 4.7 e 4.8 apresentam os gráficos de evolução da energia média do erro  $\xi_{med}(n)$  para os subconjuntos  $A_T$  e  $A_v$  em função das  $n$ -ésimas iterações ou épocas. Ressalta-se que os valores das energias foram esboçadas a cada 50 épocas, a fim de facilitar a visualização e análise.

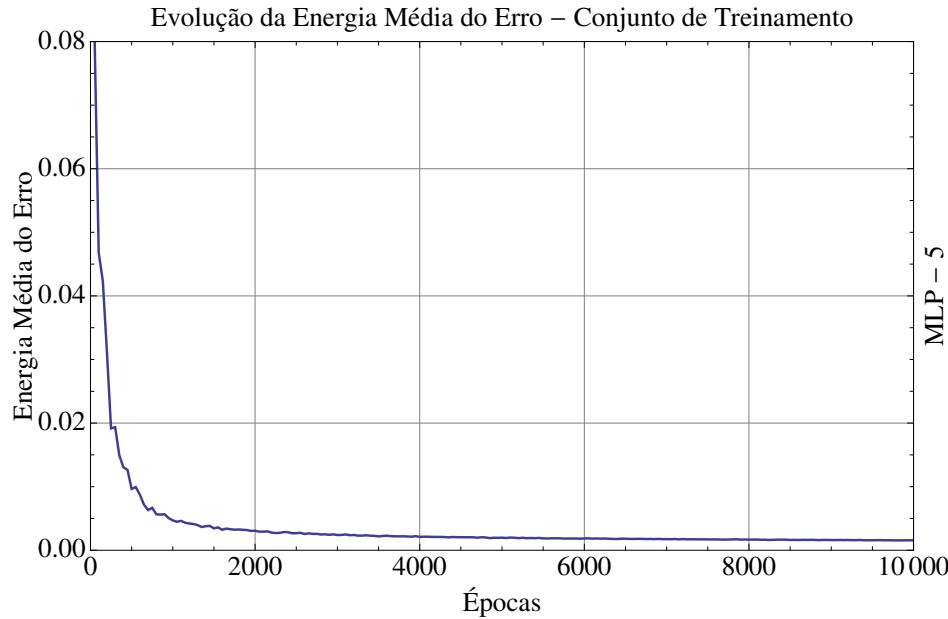


Figura 4.7: Evolução da energia média do erro do subconjunto  $A_T$  ao longo das épocas.

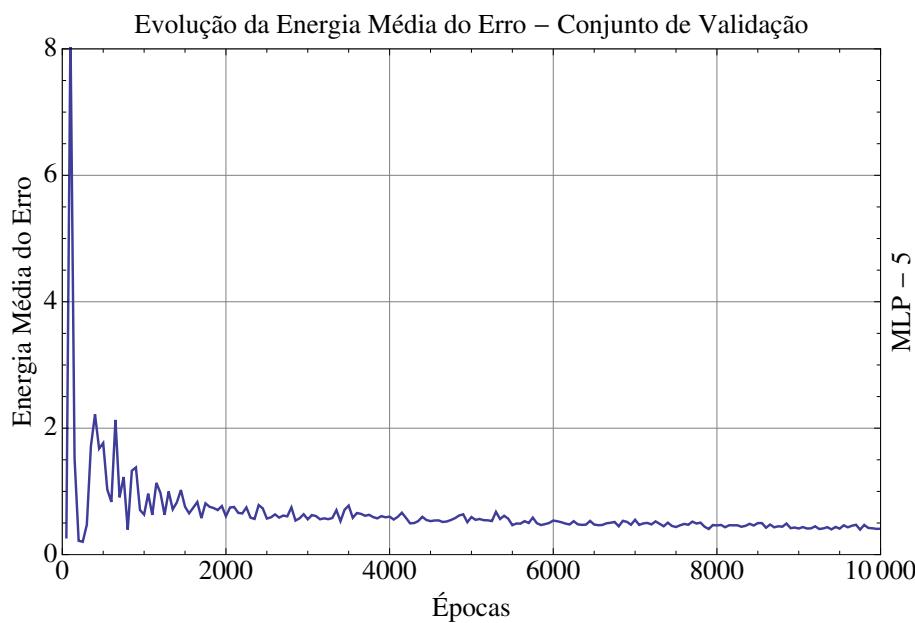


Figura 4.8: Evolução da energia média do erro do subconjunto  $A_v$  ao longo das épocas.

Observa-se que o valor médio da energia do erro  $\xi_{med}(n)$  do subconjunto  $A_T$  assumiu valores cada vez menores ao longo das épocas. Porém, é notável que a taxa de decaimento foi relativamente alta no ínicio do treinamento, em oposição ao final, no qual observa-se um decaimento bem mais lento. O mesmo pode-se dizer do valor médio da energia do erro do subconjunto  $A_v$ , o qual apresenta ligeira oscilação nas primeiras épocas, estabilizando-se em decaimento nas últimas. Contudo, não há aumento do valor de  $\xi_{med}(n)$  em  $A_v$ , o que poderia indicar *overfitting*. Assim sendo, conclui-se que o treinamento pode estar adequado para os casos de  $A_T$  e  $A_v$ . Porém, para analisar o comportamento real da *MLP* quanto à generalização, os casos de  $A_{te}$  foram testados após o treinamento. A Tabela 4.1 apresenta os valores das energias médias do erro  $\xi_{med}$  e os desvios padrão  $\sigma$  ao final do treinamento para os três subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ :

Subconjunto	$A_T$	$A_v$	$A_{te}$
$\xi_{med}$	0,00064	0,01325	0,00970
$\sigma$	0,00059	0,03185	0,01364

Tabela 4.1: Energia média do erro para os subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ .

Na Tabela 4.1, é possível observar que o subconjunto de teste  $A_{te}$  possui valor da energia média do erro menor em relação ao subconjunto de validação  $A_v$ . O valor do desvio padrão  $\sigma$  de  $A_{te}$  também é menor em relação a  $A_v$ . Isso indica que a *MLP* apresenta um bom comportamento com relação a generalização dos casos.

### 4.2.2 Arquitetura 2: MLP-10

As Figuras 4.9 e 4.10 apresentam os gráficos de evolução da energia média do erro  $\xi_{med}(n)$  para os subconjuntos  $A_T$  e  $A_v$  em função das  $n$ -ésimas iterações ou épocas.

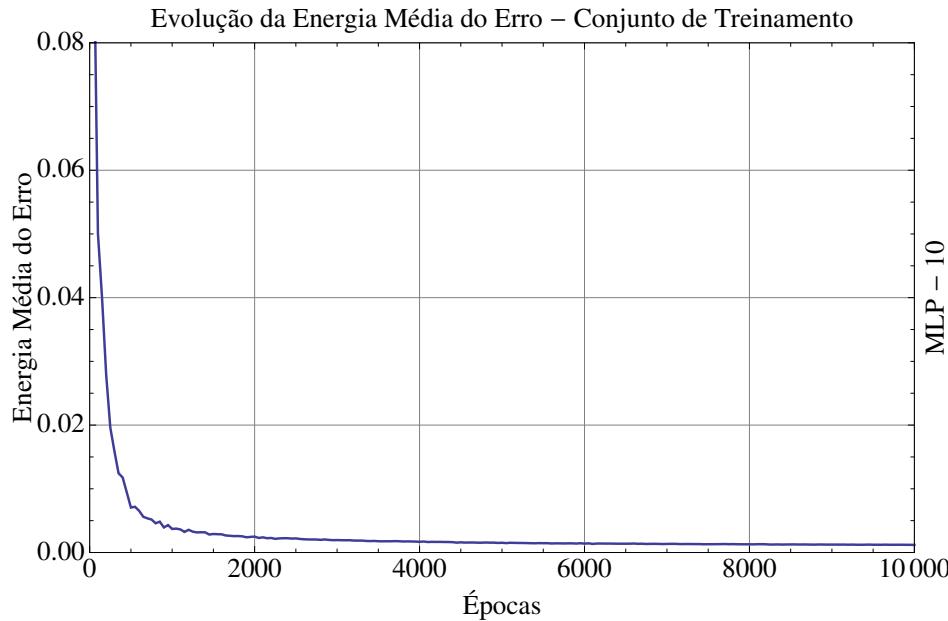


Figura 4.9: Evolução da energia média do erro do subconjunto  $A_T$  ao longo das épocas.

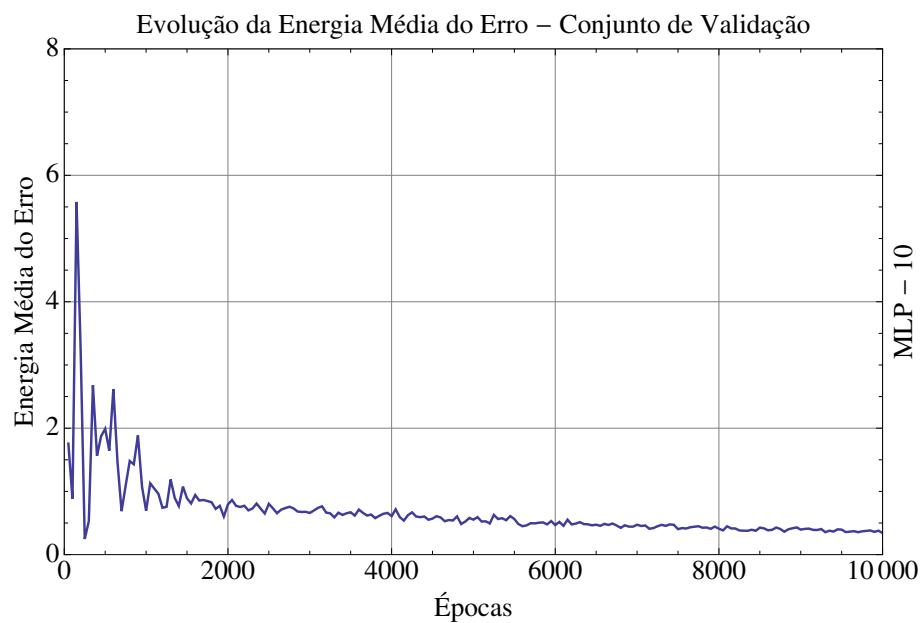


Figura 4.10: Evolução da energia média do erro do subconjunto  $A_v$  ao longo das épocas.

Nota-se que as energias médias  $\xi_{med}(n)$  dos subconjuntos  $A_T$  e  $A_v$  mantiveram um comportamento semelhante à *MLP-5*: ambas curvas decaíram com número de épocas e o subconjunto  $A_v$  também apresentou oscilação nas primeiras iterações. Da mesma forma que observado em 4.2.1, não há evidências da ocorrência de *overfitting* no processo de treinamento, já que a energia média de  $A_v$  decaiu ao longo das épocas. Desta forma, conclui-se que o treinamento pode estar adequado para os casos de  $A_T$  e  $A_v$ , sendo necessário porém testar a *MLP* para os casos de  $A_{te}$ . A Tabela 4.2 apresenta os valores das energias médias do erro  $\xi_{med}$  e os desvios padrão  $\sigma$  ao final do treinamento para os três subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ .

Subconjunto	$A_T$	$A_v$	$A_{te}$
$\xi_{med}$	0,00052	0,01521	0,01209
$\sigma$	0,00037	0,03630	0,02010

Tabela 4.2: Energia média do erro para os subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ .

Na Tabela 4.2, observa-se que o subconjunto de teste  $A_{te}$  possui valor da energia média do erro menor em relação ao subconjunto de validação  $A_v$ . O valor do desvio padrão  $\sigma$  de  $A_{te}$  também é menor em relação a  $A_v$ , à semelhança da *MLP-5*, indicando que a *MLP* apresenta um bom comportamento com relação à generalização.

### 4.2.3 Arquitetura 3: MLP-15

As Figuras 4.11 e 4.12 apresentam os gráficos de evolução da energia média do erro  $\xi_{med}(n)$  para os subconjuntos  $A_T$  e  $A_v$ , em função das  $n$ -ésimas iterações ou épocas.

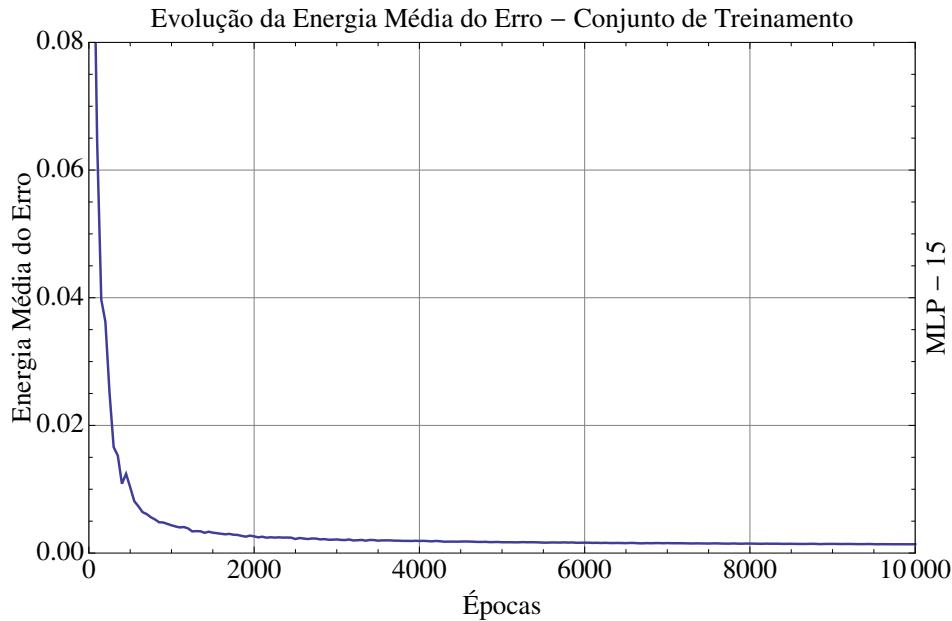


Figura 4.11: Evolução da energia média do erro do subconjunto  $A_T$  ao longo das épocas.

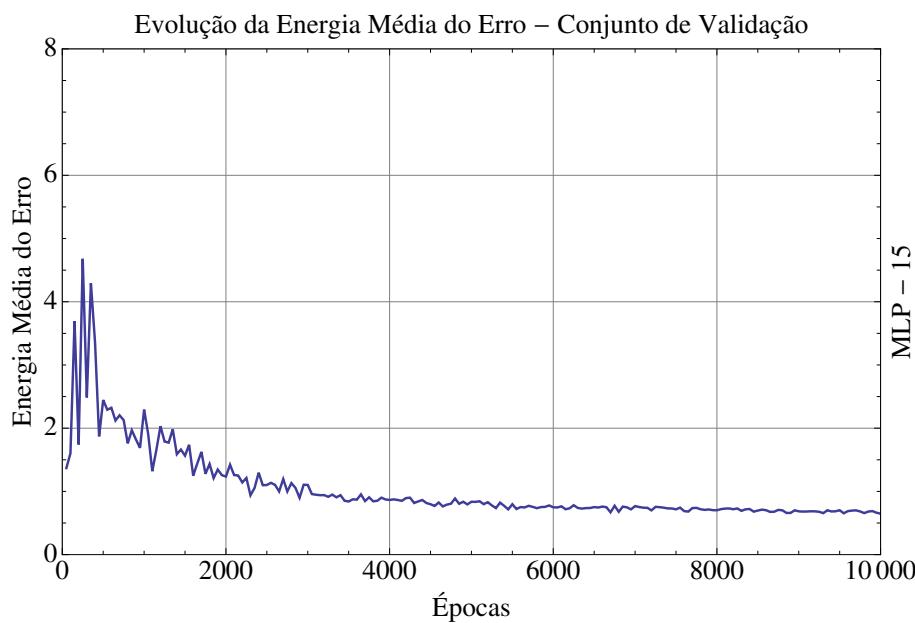


Figura 4.12: Evolução da energia média do erro do subconjunto  $A_v$  ao longo das épocas.

As energias médias  $\xi_{med}(n)$  dos subconjuntos  $A_T$  e  $A_v$ , como as Figuras 4.11 e 4.12 apresentam, mantiveram um comportamento semelhante ao obtido nos treinamentos da *MLP-5* e da *MLP-10*: ambas curvas decaíram com número de épocas e ainda ocorreu oscilação (nas primeiras épocas) na curva do subconjunto  $A_v$ . Da mesma forma que observado em 4.2.1 e em 4.2.2, não há evidências da ocorrência de *overfitting* no processo de treinamento, já que a energia média de  $A_v$  decaiu ao longo do processo de aprendizagem. Conclui-se que o treinamento está adequado para os casos de  $A_T$  e  $A_v$ , sendo necessário contudo testar a *MLP* para os casos de  $A_{te}$  a fim de verificar a generalização da rede. A Tabela 4.3 apresenta os valores das energias médias do erro  $\xi_{med}$  e os desvios padrão  $\sigma$  ao final do processo de treinamento para os três subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ .

Subconjunto	$A_T$	$A_v$	$A_{te}$
$\xi_{med}$	0,00056	0,01789	0,01175
$\sigma$	0,00040	0,04643	0,01534

Tabela 4.3: Energia média do erro para os subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ .

Observa-se na Tabela 4.3 que o subconjunto de teste  $A_{te}$  possui valor da energia média do erro menor em relação ao subconjunto de validação  $A_v$ . O valor do desvio padrão  $\sigma$  de  $A_{te}$  também é menor em relação a  $A_v$ , à semelhança do observado nos treinamentos da *MLP-5* e da *MLP-10*. Assim, esses resultados indicam que a *MLP* apresenta comportamento adequado com relação à generalização dos casos. Na Seção 4.2.4, são apresentados os valores da energia total  $\xi(k)$  para os  $k$ -ésimos casos de  $A$ , para as três *MLPs*, permitindo a visualização e comparação dos resultados de treinamento de cada rede neural.

#### 4.2.4 Comparação

A fim de observar e comparar os resultados obtidos no processo de treinamento de cada arquitetura, gerou-se um gráfico contendo todos os valores da energia total do erro  $\xi(k)$  para cada  $k$ -ésimo caso do conjunto de treinamento  $A$ , para as três *MLPs*. Como os casos de  $A$  são numerados, facilmente se distinguem no gráfico quais casos pertencem a cada um dos subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ , conforme definidos em 3.5.1.

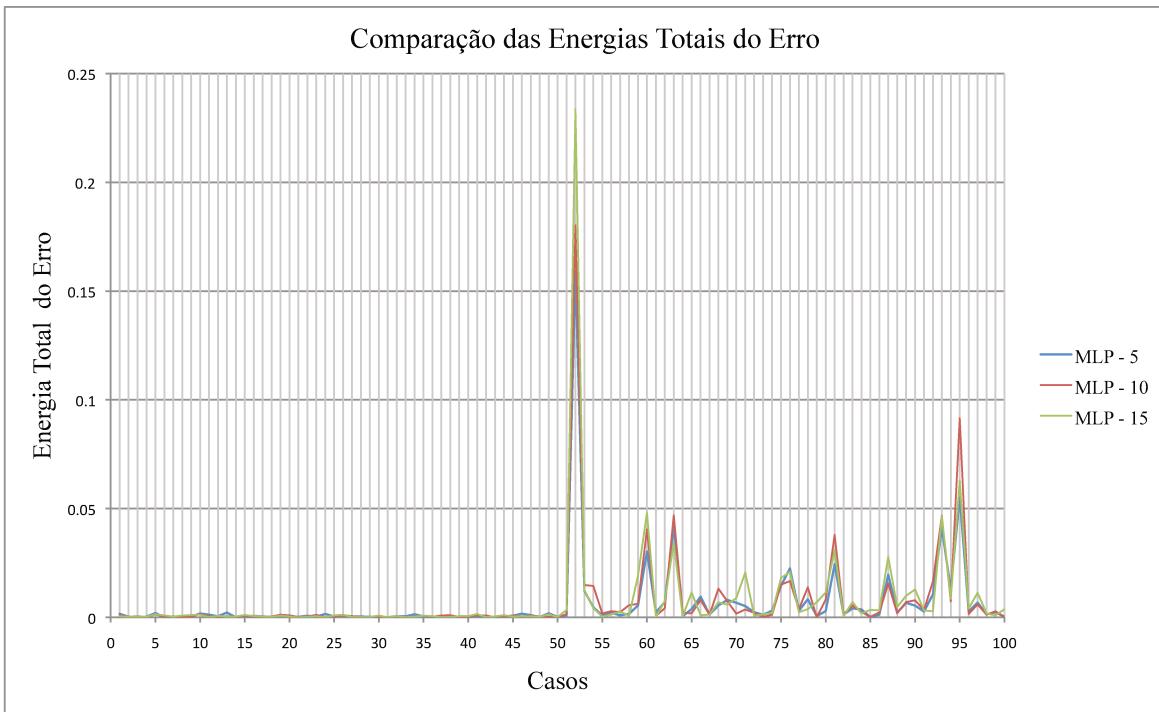


Figura 4.13: Comparação das energias totais dos erros do conjunto  $A$  para as três *MLPs*.

Pode-se observar que, para os 50 primeiros casos, correspondentes ao subconjunto  $A_T$ , as energias totais para as três *MLPs* mantiveram-se relativamente próximas e em valores razoavelmente baixos, conforme observado nas Tabelas 4.1, 4.2 e 4.3. Para os 25 casos seguintes, correspondentes ao subconjunto  $A_v$ , observa-se maior oscilação entre os valores da energia, sendo a *MLP-5* a que apresentou menor oscilação, em oposição a *MLP-15*, que oscilou com maior amplitude. Para os últimos 25 casos, referentes ao subconjunto  $A_{te}$ , a *MLP-10* apresentou maiores amplitudes do erro em alguns casos, enquanto que a *MLP-15* gerou maiores erros em outros. A *MLP-5*, para esse conjunto, manteve-se com menor valor da energia total do erro em relação às outras duas. Ressalta-se também que as distribuições dos erros totais das três *MLPs* tenderam à mesma variação, porém com amplitudes diferentes. Para quantificar essas diferenças observadas, elencou-se em tabelas os

valores da energia média e do desvio padrão para cada um dos conjuntos, para as três redes neurais. As Tabelas 4.4, 4.5 e 4.6 apresentam os resultados para os três subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ .

Arquiteturas	$A_T$		
	<i>MLP-5</i>	<i>MLP-10</i>	<i>MLP-15</i>
$\xi_{med}$	0,00064	0,00052	0,00056
$\sigma$	0,00059	0,00037	0,00040

Tabela 4.4: Energia média do erro para o subconjunto  $A_T$ .

Arquiteturas	$A_v$		
	<i>MLP-5</i>	<i>MLP-10</i>	<i>MLP-15</i>
$\xi_{med}$	0,01325	0,01521	0,01789
$\sigma$	0,03185	0,03630	0,04643

Tabela 4.5: Energia média do erro para o subconjunto  $A_v$ .

Arquiteturas	$A_{te}$		
	<i>MLP-5</i>	<i>MLP-10</i>	<i>MLP-15</i>
$\xi_{med}$	0,00970	0,01209	0,01175
$\sigma$	0,01364	0,02010	0,01534

Tabela 4.6: Energia média do erro para o subconjunto  $A_{te}$ .

Observa-se na Tabela 4.4 que as arquiteturas *MLP-10* e *MLP-15* apresentam menores valores de energia média do erro. Por essa observação, pode-se afirmar que elas possuem melhor aproximação da função original. Porém, observa-se através das Tabelas 4.5 e 4.6, que a *MLP-5* é a que apresenta menor valor tanto para a energia média quanto para o desvio padrão nos casos de  $A_v$  e  $A_{te}$ , refutando a afirmação anterior. Isso reforça a conclusão obtida com a observação do gráfico da Figura 4.13. Assim, a rede neural que apresenta melhor desempenho com relação à generalização é a *MLP-5*. Essa conclusão pode estar relacionada aos efeitos observados do superdimensionamento, descritos em Braga *et al.* (2007). Com o aumento do número de neurônios, o percentual de soluções boas é cada vez menor em relação ao número de todas as outras soluções possíveis. Isso pode fazer

com que o algoritmo de aprendizado, em sua busca estocástica pela solução, escolha qualquer uma delas como satisfatórias segundo o critério de minimização de erro (Braga *et al.*, 2007), sem, contudo, satisfazer necessariamente uma melhor generalização para os casos de validação e de teste. Assim, a *MLP-5*, pelos dados do conjunto  $A$ , parece fornecer uma solução melhor do que as outras duas arquiteturas, sob essa ótica do superdimensionamento.

## 4.3 Redes Neurais e o Modelo Unidimensional

Esta seção tem o objetivo de apresentar os resultados do arranjo das redes neurais com o modelo unidimensional do acoplamento poço-reservatório. São apresentados os resultados obtidos com o arranjo das *MLPs* treinadas com o modelo e os erros em norma  $L^2$  para as curvas de fluxo  $Q_{hw}(x)$ ,  $p(x)$ ,  $\frac{dQ_{hw}(x)}{dx}$  e  $\frac{dp(x)}{dx}$ . Também são apresentados os erros das vazões finais obtidas em cada arranjo. Ao final, uma comparação entre as três arquiteturas é realizada a fim de avaliar o comportamento delas quando aplicadas na resolução das equações do modelo simplificado.

### 4.3.1 *MLP-5* e Modelo Unidimensional

As Tabelas 4.7, 4.8, 4.9 e 4.10 apresentam os valores médios e os desvios padrão dos erros em  $L^2$  para os três subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ , para cada uma das curvas correspondentes ao fluxo no interior do poço. Ressalta-se que os valores dos erros e desvios padrão foram normalizados a partir de um valor médio da norma  $L^2$  das curvas considerando todos os casos do conjunto  $A$ . Essa normalização foi realizada com objetivo de adimensionalizar os valores dos erros e desvios padrão.

Subconjunto	$L^2 - Q_{hw}(x)$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,00729	0,01684	0,01459
$\sigma$	0,00824	0,02432	0,01733

Tabela 4.7: Norma  $L^2$  média do erro e desvio padrão - curva  $Q_{hw}(x)$ .

Subconjunto	$L^2 - p(x)$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	$7,24E - 6$	$8,19E - 6$	$7,68E - 6$
$\sigma$	$3,77E - 6$	$3,63E - 6$	$4,17E - 6$

Tabela 4.8: Norma  $L^2$  média do erro e desvio padrão - curva  $p(x)$ .

	$L^2 - dQ_{hw}(x)/dx$		
Subconjunto	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,02222	0,06796	0,02353
$\sigma$	0,02745	0,12939	0,02745

Tabela 4.9: Norma  $L^2$  média do erro e desvio padrão - curva  $dQ_{hw}(x)/dx$ .

	$L^2 - dp(x)/dx$		
Subconjunto	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,00181	0,01595	0,01878
$\sigma$	0,00393	0,01859	0,02200

Tabela 4.10: Norma  $L^2$  média do erro e desvio padrão - curva  $dp(x)/dx$ .

Pode-se observar, pelas Tabelas 4.7 e 4.9, que os valores médios do erro em norma  $L^2$  e do desvio padrão foram menores para o subconjunto  $A_{te}$  em relação ao subconjunto  $A_v$ . Na Tabela 4.8, observa-se que o valor médio do erro em  $L^2$  é menor para  $A_{te}$ , porém o desvio padrão é maior em relação ao subconjunto  $A_v$ . Pela Tabela 4.10, é possível notar que o valor médio do erro e do desvio padrão são menores para o conjunto de validação,  $A_v$ . Esses resultados são semelhantes aos obtidos no treinamento da *MLP-5* (Subseção 4.2.1), a menos da distribuição do erro da curva  $\frac{dp(x)}{dx}$ . Uma explicação para essa diferença é o fato de que as medidas dos erros são diferentes entre as usadas para análise do desempenho das *MLPs* e as usadas nas curvas de fluxo.

A Tabela 4.11 apresenta a média do erro em norma  $L^1$  alterada da vazão total  $Q_{hw}$  e o desvio padrão para os subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ .

	$L^1 - Q_{hw}$		
Subconjunto	$A_T$	$A_v$	$A_{te}$
$L^1$ média	0,615	1,362	1,560
$\sigma$	0,338	0,828	1,581

Tabela 4.11: Norma  $L^1$  (alterada) média do erro e desvio padrão -  $Q_{hw}$ .

Observa-se que os maiores valores do erro e do desvio padrão ocorrem para o subconjunto  $A_{te}$ .

### 4.3.2 MLP-10 e Modelo Unidimensional

As Tabelas 4.12, 4.13, 4.14 e 4.15 apresentam os valores médios e os desvios padrão dos erros em  $L^2$  para os três subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ , para todas as curvas de fluxo do poço. Destaca-se que os valores foram normalizados, assim como realizado na Subseção 4.3.1.

Subconjunto	$L^2 - Q_{hw}(x)$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,00729	0,01449	0,01373
$\sigma$	0,00824	0,01980	0,01343

Tabela 4.12: Norma  $L^2$  média do erro e desvio padrão - curva  $Q_{hw}(x)$ .

Subconjunto	$L^2 - p(x)$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	$7,14E - 6$	$8,19E - 6$	$7,43E - 6$
$\sigma$	$3,77E - 6$	$3,67E - 6$	$4,25E - 6$

Tabela 4.13: Norma  $L^2$  média do erro e desvio padrão - curva  $p(x)$ .

Subconjunto	$L^2 - dQ_{hw}(x)/dx$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,02222	0,07058	0,03137
$\sigma$	0,02745	0,12286	0,03398

Tabela 4.14: Norma  $L^2$  média do erro e desvio padrão - curva  $dQ_{hw}(x)/dx$ .

Subconjunto	$L^2 - dp(x)/dx$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,00181	0,01428	0,01949
$\sigma$	0,00393	0,01508	0,02270

Tabela 4.15: Norma  $L^2$  média do erro e desvio padrão - curva  $dp(x)/dx$ .

Observa-se nas Tabelas 4.12 e 4.14 que os valores médios do erro em  $L^2$  e do desvio padrão foram menores para o subconjunto  $A_{te}$  em relação ao subconjunto  $A_v$ , à semelhança do que fora observado na Subseção 4.3.1. O valor médio do erro em  $L^2$  para curva  $p(x)$  é menor para o subconjunto  $A_{te}$ , porém o desvio padrão é maior em relação ao subconjunto  $A_v$ , conforme observado na Tabela 4.13. Na Tabela 4.15, observa-se que os valores médios de erro em  $L^2$  e do desvio padrão para a curva  $\frac{dp(x)}{dx}$  são menores para o conjunto de validação,  $A_v$ . Essas observações são semelhantes aos resultados apresentados na Subseção 4.3.1.

A Tabela 4.16 apresenta a média do erro em norma  $L^1$  alterada da vazão total  $Q_{hw}$  e o desvio padrão para os três subconjuntos,  $A_T$ ,  $A_v$  e  $A_{te}$ .

Subconjunto	$L^1 - Q_{hw}$		
	$A_T$	$A_v$	$A_{te}$
$L^1$ média	0,615	1,191	1,689
$\sigma$	0,338	0,980	1,481

Tabela 4.16: Norma  $L^1$  (alterada) média do erro e desvio padrão -  $Q_{hw}$ .

Nota-se, pela Tabela 4.16, que os maiores valores médios do erro e do desvio padrão ocorrem para o subconjunto  $A_{te}$ .

### 4.3.3 MLP-15 e Modelo Unidimensional

As Tabelas 4.17, 4.18, 4.19 e 4.20 apresentam os valores médios e os desvios padrão dos erros em  $L^2$  para os três subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ , para as curvas correspondentes ao fluxo no interior do poço. Os valores foram normalizados, conforme realizado nas subseções anteriores.

Subconjunto	$L^2 - Q_{hw}(x)$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,00729	0,01624	0,01670
$\sigma$	0,00824	0,01903	0,01728

Tabela 4.17: Norma  $L^2$  média do erro e desvio padrão - curva  $Q_{hw}(x)$ .

Subconjunto	$L^2 - p(x)$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	$7,14E - 6$	$8,10E - 6$	$7,59E - 6$
$\sigma$	$3,77E - 6$	$3,47E - 6$	$4,17E - 6$

Tabela 4.18: Norma  $L^2$  média do erro e desvio padrão - curva  $p(x)$ .

Subconjunto	$L^2 - dQ_{hw}(x)/dx$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,02222	0,07058	0,03398
$\sigma$	0,02745	0,12024	0,02875

Tabela 4.19: Norma  $L^2$  média do erro e desvio padrão - curva  $dQ_{hw}(x)/dx$ .

Subconjunto	$L^2 - dp(x)/dx$		
	$A_T$	$A_v$	$A_{te}$
$L^2$ média	0,00181	0,01437	0,02490
$\sigma$	0,00393	0,01651	0,02870

Tabela 4.20: Norma  $L^2$  média do erro e desvio padrão - curva  $dp(x)/dx$ .

Observa-se, na Tabela 4.17, que o erro médio em  $L^2$  da curva  $Q_{hw}(x)$  para o subconjunto  $A_{te}$  foi ligeiramente maior em relação ao subconjunto  $A_v$ , ocorrendo o oposto com o valor do desvio padrão. A Tabela 4.19 indica que os valores médios do erro e do desvio padrão foram menores para o subconjunto  $A_{te}$ . Pela Tabela 4.18, observa-se que o valor médio do erro da curva  $p(x)$  é menor para  $A_{te}$ , porém o desvio padrão é maior em relação ao subconjunto  $A_v$ . Os valores médios de erro e do desvio padrão da curva  $\frac{dp(x)}{dx}$  são menores para o conjunto de validação,  $A_v$ . Esses resultados assemelham-se aos apresentados nas Subseções 4.3.1 e 4.3.2, a menos do erro da curva  $Q_{hw}(x)$ .

A Tabela 4.21 apresenta a média do erro em norma  $L^1$  alterada da vazão total  $Q_{hw}$  e o desvio padrão para os subconjuntos  $A_T$ ,  $A_v$  e  $A_{te}$ .

Subconjunto	$L^1 - Q_{hw}$		
	$A_T$	$A_v$	$A_{te}$
$L^1$ média	0,615	1,637	1,905
$\sigma$	0,338	1,314	1,746

Tabela 4.21: Norma  $L^1$  (alterada) média do erro e desvio padrão -  $Q_{hw}$ .

Observa-se que os maiores valores médios do erro e do desvio padrão ocorrem para o subconjunto  $A_{te}$ , à semelhança do observado em 4.3.1 e em 4.3.2.

#### 4.3.4 Comparação

A fim de comparar o desempenho de cada *MLP* com relação à aplicação no modelo do acoplamento poço-reservatório, foram gerados gráficos contendo os erros em  $L^2$  para cada uma das curvas  $Q_{hw}(x)$ ,  $p(x)$ ,  $\frac{dQ_{hw}(x)}{dx}$  e  $\frac{dp(x)}{dx}$ , para cada caso do conjunto  $A$ . Como os casos são numerados, é possível verificar quais pertencem a cada subconjunto  $A_T$ ,  $A_v$  e  $A_{te}$ , de acordo com a definição apresentada em 3.5.1. As figuras na sequência apresentam os gráficos comparativos:

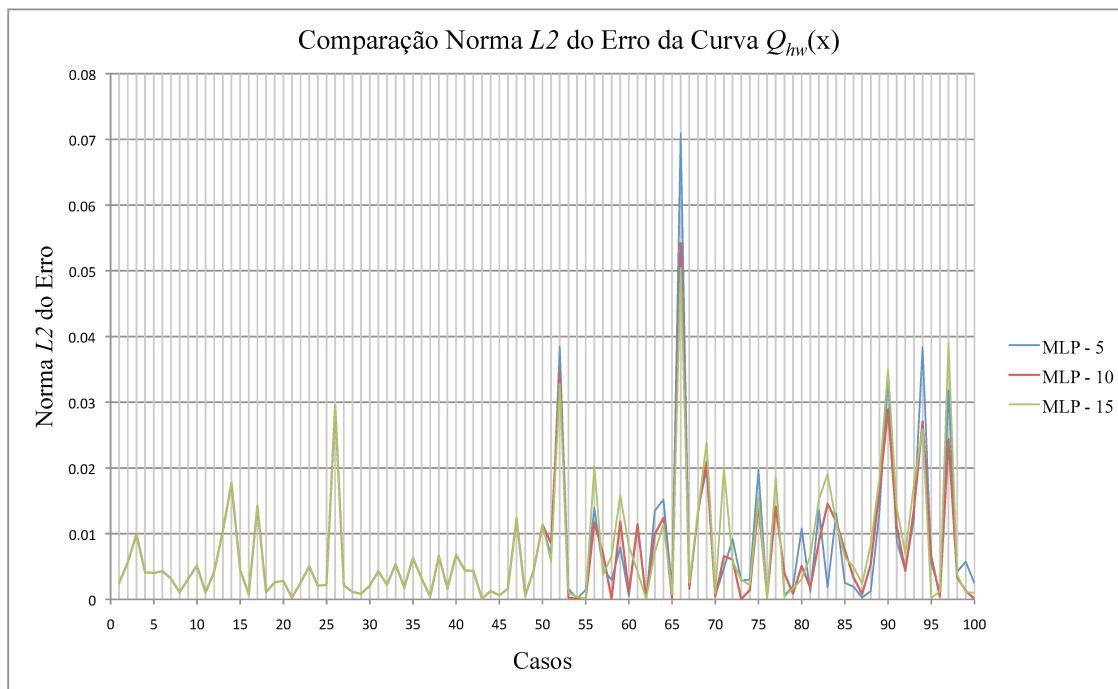


Figura 4.14: Comparação norma  $L^2$  do conjunto  $A$  para as três  $MLPs$  - curva  $Q_{hw}(x)$ .

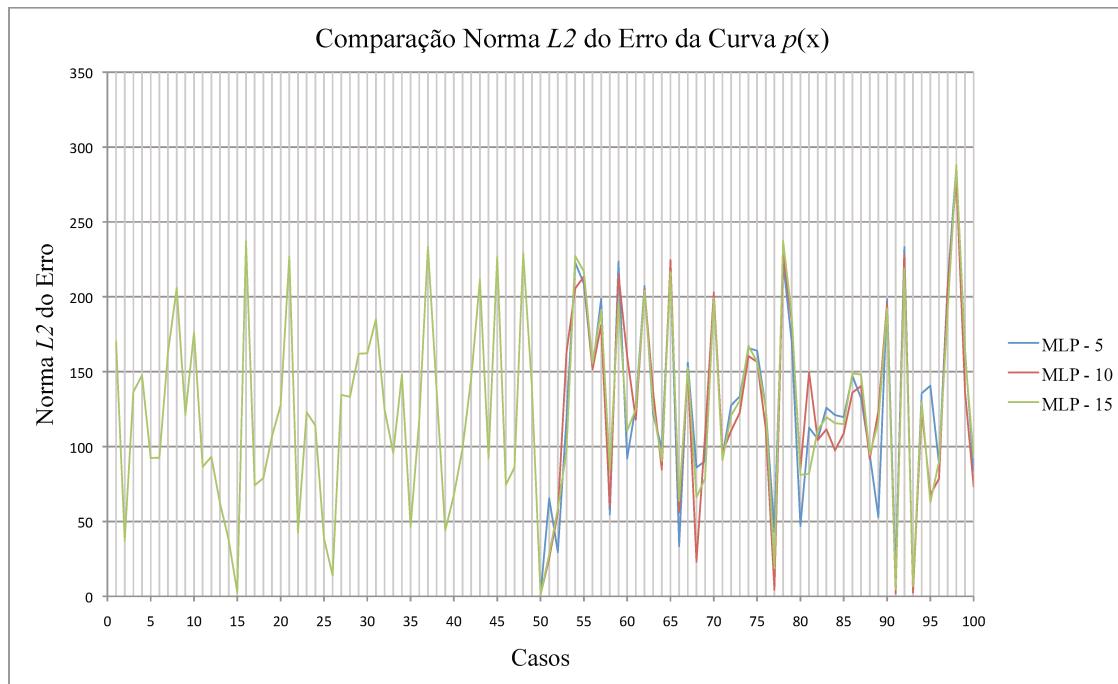


Figura 4.15: Comparação norma  $L^2$  do conjunto  $A$  para as três  $MLPs$  - curva  $p(x)$ .

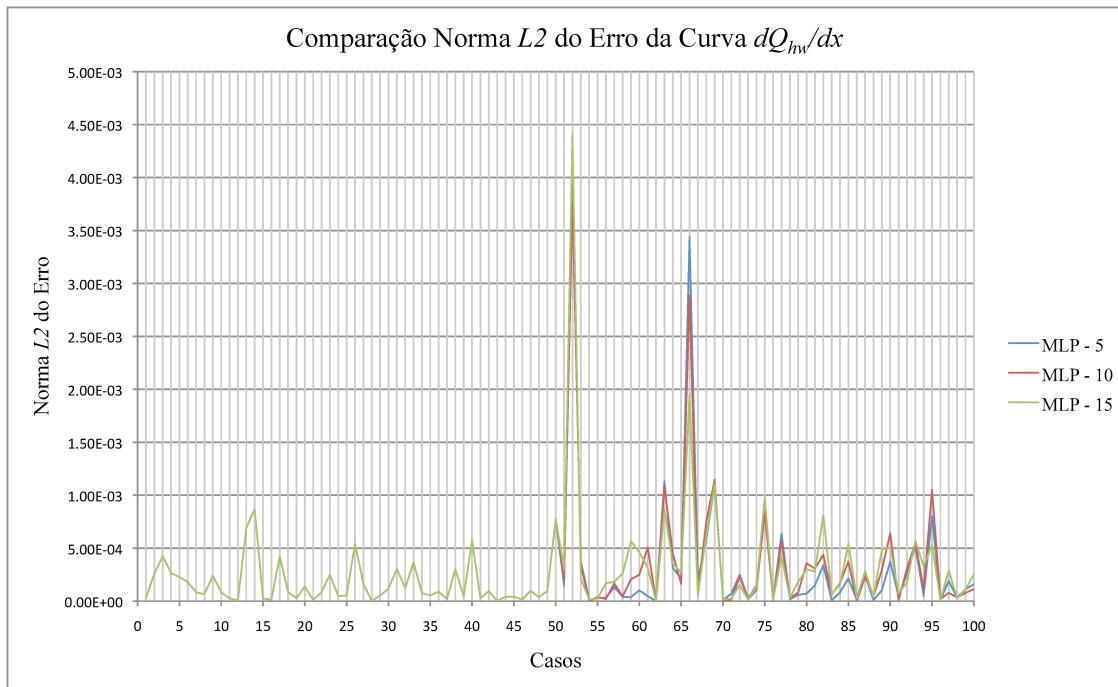


Figura 4.16: Comparação norma  $L^2$  do conjunto  $A$  para as três MLPs - curva  $dQ_{hw}(x)/dx$ .

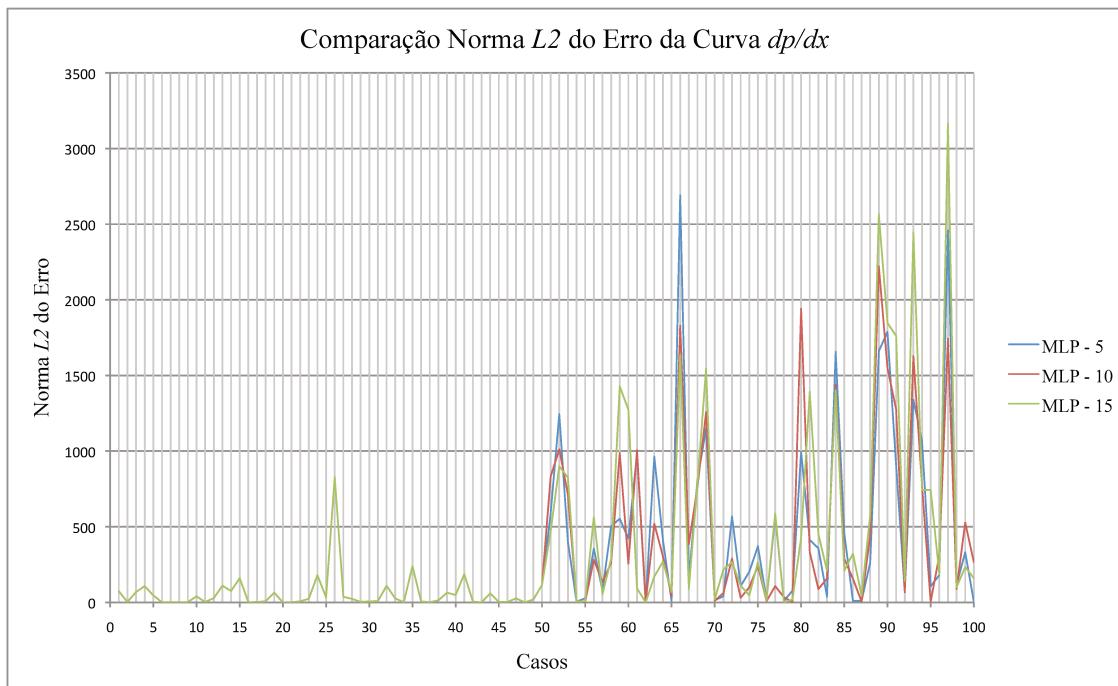


Figura 4.17: Comparação norma  $L^2$  do conjunto  $A$  para as três MLPs - curva  $dp(x)/dx$ .

Observa-se, na Figura 4.14, que os valores do erro são idênticos entre as três *MLPs* para o subconjunto  $A_T$ . Observa-se também que os erros entre as *MLPs* parecem variar nos mesmos casos dos conjuntos  $A_v$  e  $A_{te}$ , porém em amplitudes diferentes. O mesmo é observado nas outras curvas, conforme esboçam as Figuras 4.15, 4.16 e 4.17.

Para comparar os erros em  $L^1$  da vazão total  $Q_{hw}$ , a Figura 4.18 apresenta as curvas dos erros para cada caso do conjunto  $A$ , para as três *MLPs*:

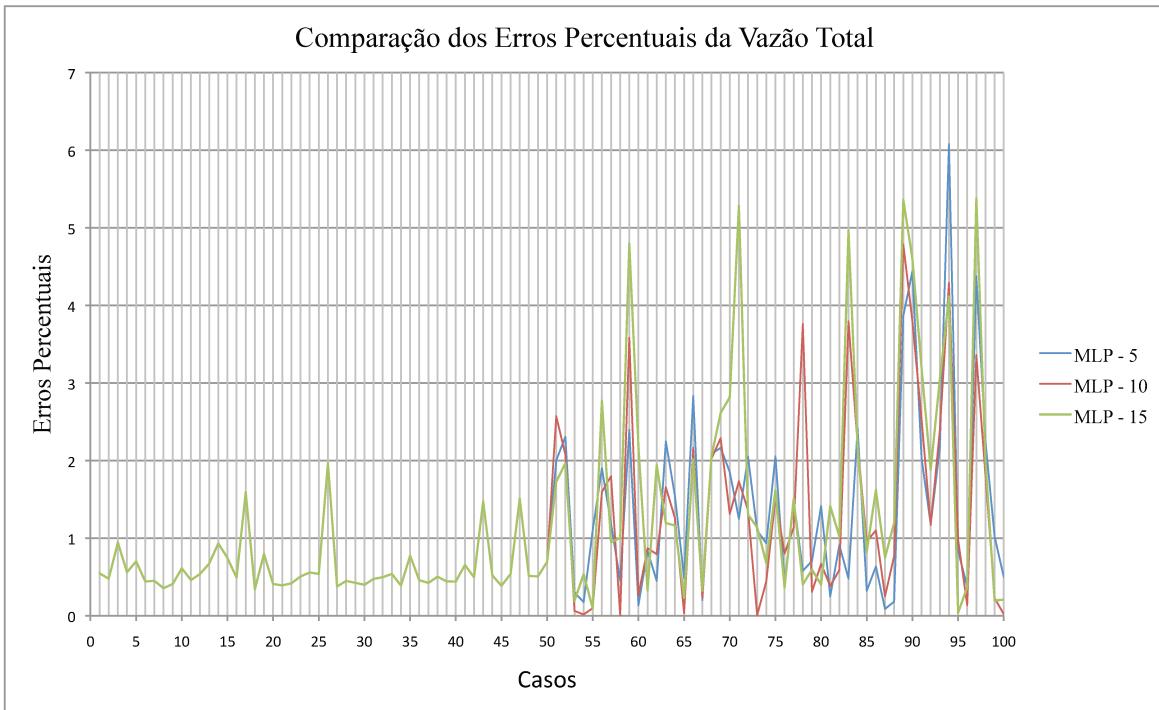


Figura 4.18: Comparação norma  $L^1$  dos erros do conjunto  $A$  para as três *MLPs* -  $Q_{hw}$ .

Nota-se, pela Figura 4.18, que as três redes neurais apresentam valores de erros semelhantes para os casos do subconjunto  $A_T$ . Porém, para os casos dos subconjuntos  $A_v$  e  $A_{te}$ , as variações dos erros das três *MLPs* não coincidem. Isso é explicado pelo fato de que os valores das vazões totais são resultados das interações das quatro curvas de fluxo,  $Q_{hw}(x)$ ,  $p(x)$ ,  $\frac{dQ_{hw}(x)}{dx}$  e  $\frac{dp(x)}{dx}$ . Assim, presume-se que pequenas diferenças observadas nas curvas entre as três redes são suficientes para ocasionar maiores diferenças nos valores das vazões totais.

A fim de se estabelecer uma comparação, em termos dos valores dos erros e de desvio padrão, resumiu-se nas Tabelas 4.22, 4.23 e 4.24, os resultados de cada arranjo *MLP*-modelo1D, para cada subconjunto de  $A$ :

		$A_T$		
Curvas	Arquiteturas	<i>MLP-5</i>	<i>MLP-10</i>	<i>MLP-15</i>
$Q_{hw}(x)$	<i>L</i> <sup>2</sup> <i>média</i>	0,00729	0,00729	0,00729
	$\sigma$	0,00824	0,00824	0,00824
$p(x)$	<i>L</i> <sup>2</sup> <i>média</i>	$7,24E - 6$	$7,24E - 6$	$7,24E - 6$
	$\sigma$	$3,77E - 6$	$3,77E - 6$	$3,77E - 6$
$dQ_{hw}(x)/dx$	<i>L</i> <sup>2</sup> <i>média</i>	0,02222	0,02222	0,02222
	$\sigma$	0,02745	0,02745	0,02745
$dp(x)/dx$	<i>L</i> <sup>2</sup> <i>média</i>	0,00181	0,00181	0,00181
	$\sigma$	0,00393	0,00393	0,00393
$Q_{hw}$	<i>L</i> <sup>1</sup> <i>média</i>	0,615	0,615	0,615
	$\sigma$	0,338	0,338	0,338

Tabela 4.22: Comparativo entre os erros das curvas e da vazão total, subconjunto  $A_T$ .

		$A_v$		
Curvas	Arquiteturas	<i>MLP-5</i>	<i>MLP-10</i>	<i>MLP-15</i>
$Q_{hw}(x)$	<i>L</i> <sup>2</sup> <i>média</i>	0,01684	0,01449	0,01624
	$\sigma$	0,02432	0,01980	0,01903
$p(x)$	<i>L</i> <sup>2</sup> <i>média</i>	$8,19E - 6$	$8,19E - 6$	$8,10E - 6$
	$\sigma$	$3,63E - 6$	$3,67E - 6$	$3,47E - 6$
$dQ_{hw}(x)/dx$	<i>L</i> <sup>2</sup> <i>média</i>	0,06796	0,07058	0,07058
	$\sigma$	0,12939	0,12286	0,12024
$dp(x)/dx$	<i>L</i> <sup>2</sup> <i>média</i>	0,01595	0,01428	0,01437
	$\sigma$	0,01859	0,01508	0,01651
$Q_{hw}$	<i>L</i> <sup>1</sup> <i>média</i>	1,363	1,191	1,637
	$\sigma$	0,828	0,980	1,314

Tabela 4.23: Comparativo entre os erros das curvas e da vazão total, subconjunto  $A_v$ .

Curvas	Arquiteturas	$A_{te}$		
		<i>MLP-5</i>	<i>MLP-10</i>	<i>MLP-15</i>
$Q_{hw}(x)$	$L^2$ média	0,01459	0,01373	0,01670
	$\sigma$	0,01733	0,01343	0,01728
$p(x)$	$L^2$ média	$7,68E - 6$	$7,43E - 6$	$7,59E - 6$
	$\sigma$	$4,17E - 6$	$4,25E - 6$	$4,17E - 6$
$dQ_{hw}(x)/dx$	$L^2$ média	0,02353	0,03137	0,03398
	$\sigma$	0,02745	0,03398	0,02875
$dp(x)/dx$	$L^2$ média	0,01878	0,01949	0,02490
	$\sigma$	0,02200	0,02270	0,02870
$Q_{hw}$	$L^1$ média	1,560	1,689	1,905
	$\sigma$	1,581	1,481	1,746

Tabela 4.24: Comparativo entre os erros das curvas e da vazão total, subconjunto  $A_{te}$ .

Nota-se, pela Tabela 4.22, que os valores dos erros e dos desvios padrão são interessantemente idênticos para todas as *MLPs*, em cada curva, comprovando o que foi observado nos gráficos anteriores. Isso pode levar à conclusão de que as *MLPs* atingiram uma configuração semelhante referente aos casos utilizados para os ajustes dos pesos sinápticos (subconjunto de treinamento). Considerando a Tabela 4.23, nota-se que os menores valores de erros e desvios padrão aparecem ora na *MLP-10*, ora na *MLP-15*. Pela Tabela 4.24, os menores valores são notados para as *MLP-5* e *MLP-10*. Essas observações podem levar à conclusão de que o projeto de uma arquitetura adequada para um problema como tal exige ponderar o comportamento das redes em todos os subconjuntos disponíveis, e não somente no subconjunto de treinamento. Para a presente análise, parece que a *MLP-10* seria a mais adequada, por apresentar em maior parte das curvas dos casos de *A* os menores erros e desvios padrão.

A fim de ilustrar os resultados em termos das topologias das curvas de fluxo entre o modelo unidimensional e o tridimensional, seguem gráficos referentes a dois casos. Optou-se em escolher dois casos que não pertencem ao subconjunto  $A_T$  e que apresentam energia de erro ( $\xi$ ) baixa e alta, representando assim casos para os quais a generalização da *MLP* foi boa e ruim, respectivamente. Dessa forma, a partir do gráfico da Figura 4.13, observa-se que o caso 52, para *MLP-15*, apresenta um salto relativamente grande no valor da energia total. Optou-se assim em escolhê-lo como um representante para o qual a generalização apresentou maior distorção. Em oposição, optou-se no caso 100, *MLP-10*, como um caso em que a generalização apresentou-se boa. Seguem as figuras referentes à comparação topológica das curvas de fluxo provenientes do modelo de elementos finitos e do modelo unidimensional:

- Exemplo de boa generalização, caso 100, *MLP-10*.

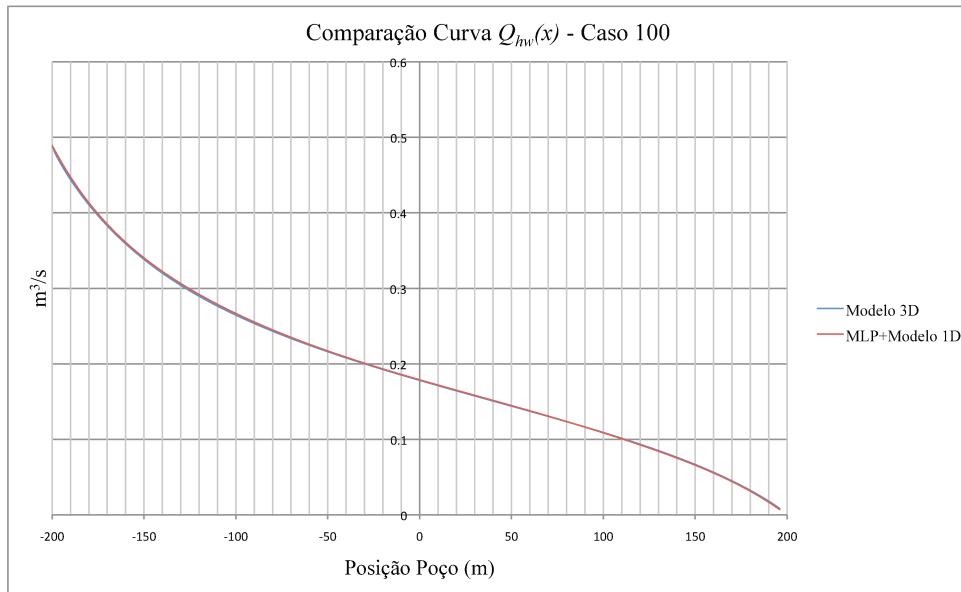


Figura 4.19: Comparação modelo 3D e *MLP+modelo 1D*, caso 100 - curva  $Q_{hw}(x)$ .

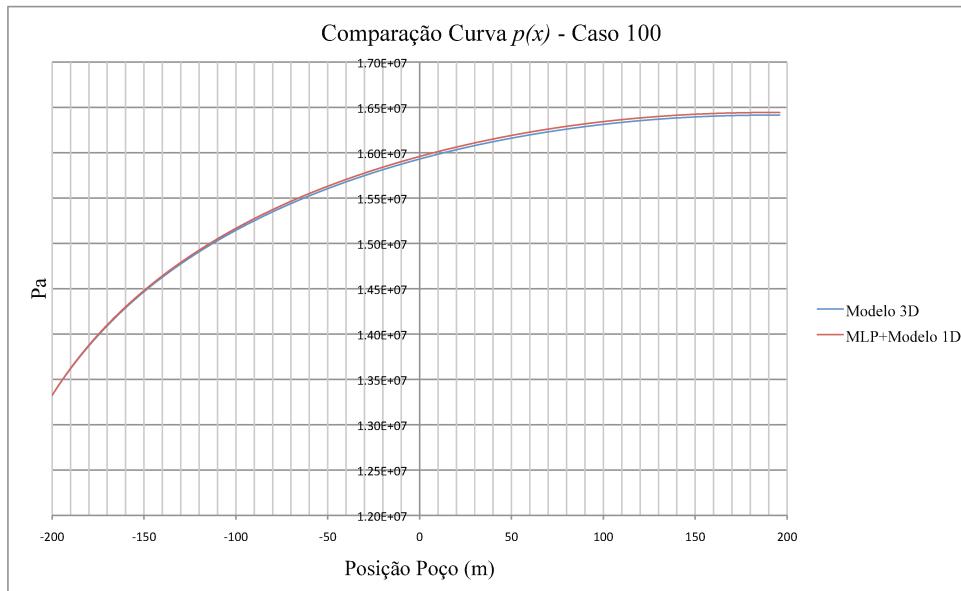


Figura 4.20: Comparação modelo 3D e *MLP+modelo 1D*, caso 100 - curva  $p(x)$ .

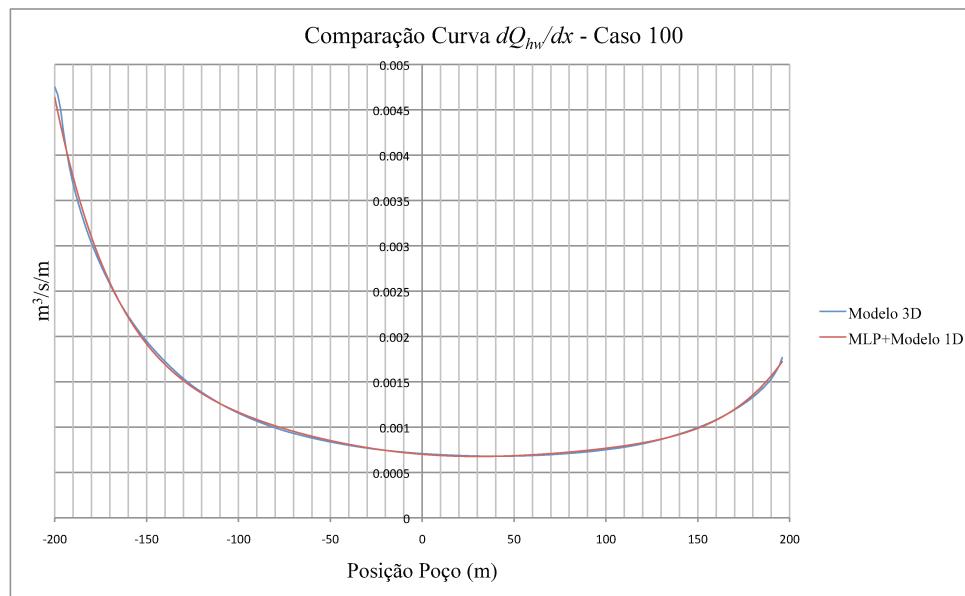


Figura 4.21: Comparação modelo 3D e *MLP+modelo 1D*, caso 100 - curva  $dQ_{hw}(x)/dx$ .

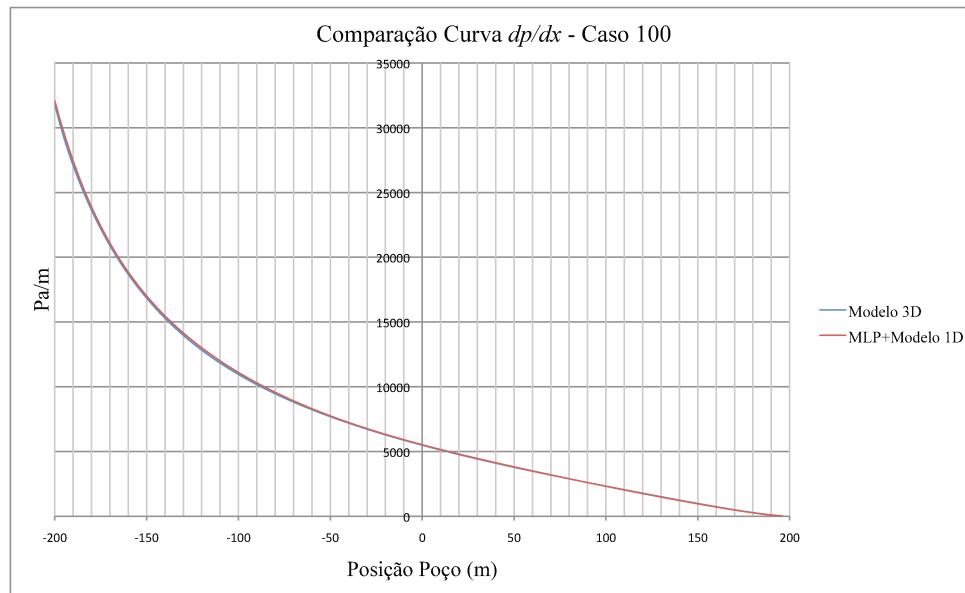


Figura 4.22: Comparação modelo 3D e *MLP+modelo 1D*, caso 100 - curva  $dp(x)/dx(x)$ .

- Exemplo de generalização ruim, caso 52, *MLP-15*.

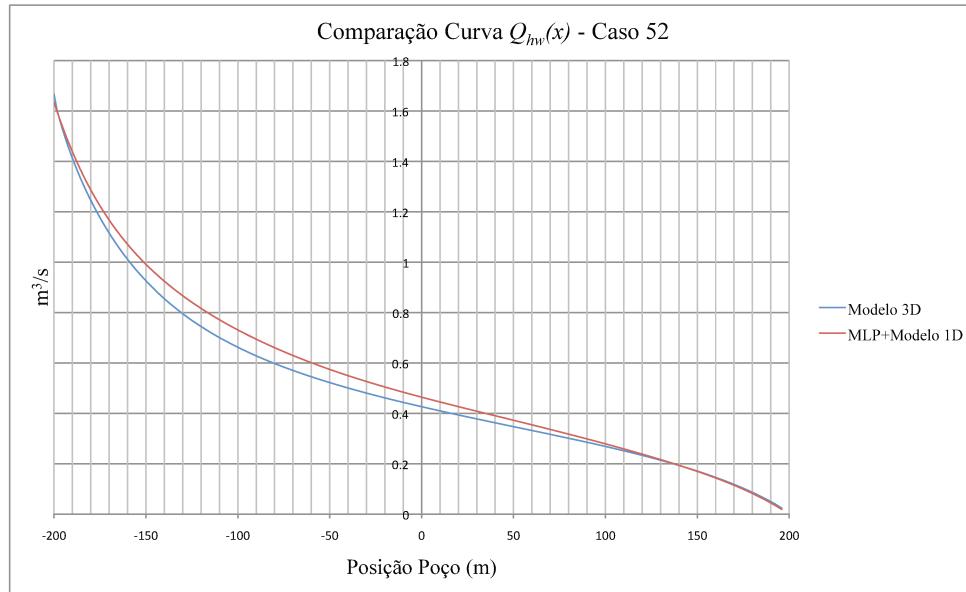


Figura 4.23: Comparação modelo 3D e *MLP+modelo 1D*, caso 52 - curva  $Q_{hw}(x)$ .

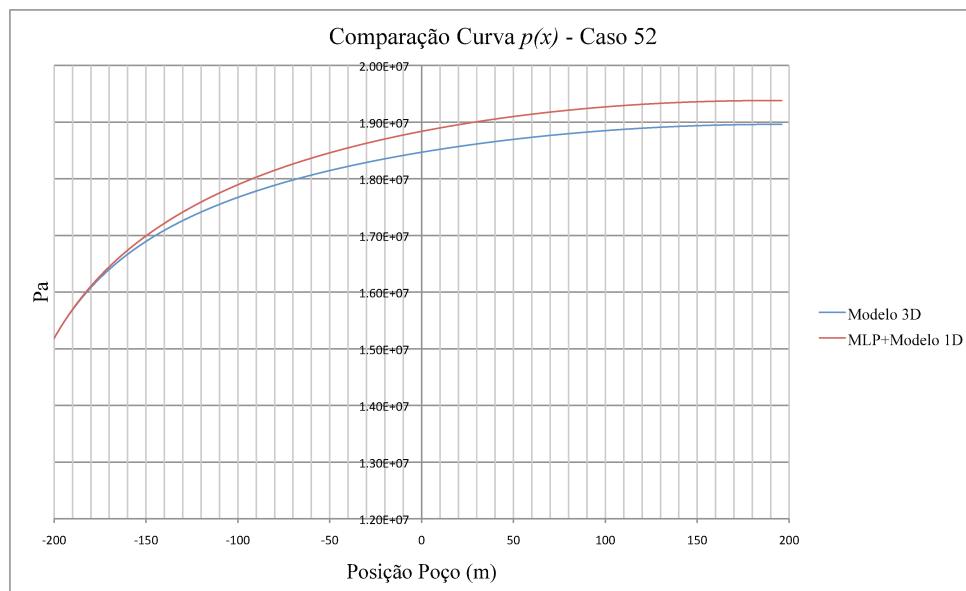


Figura 4.24: Comparação modelo 3D e *MLP+modelo 1D*, caso 52 - curva  $p(x)$ .

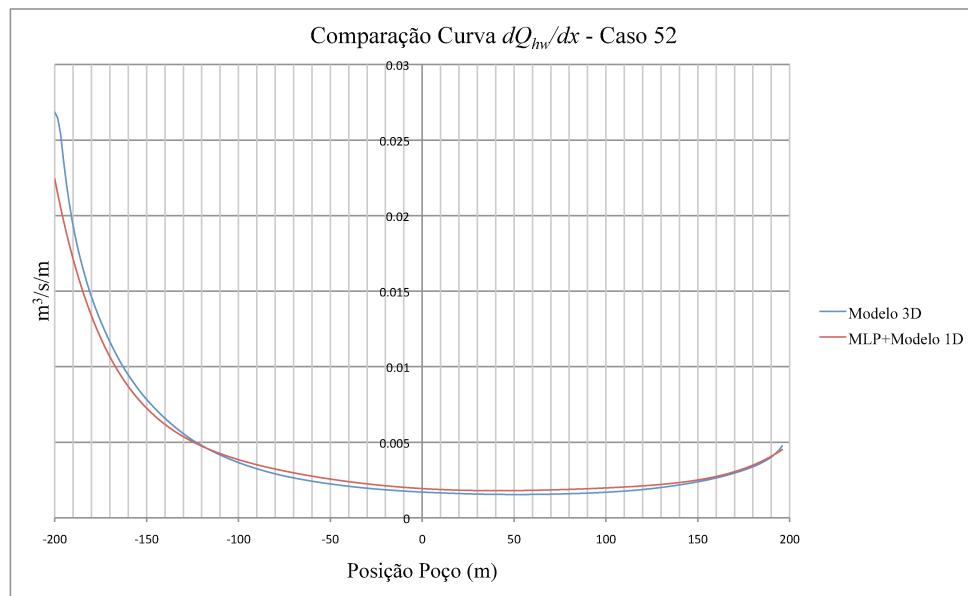


Figura 4.25: Comparação modelo 3D e *MLP+modelo 1D*, caso 52 - curva  $dQ_{hw}(x)/dx$ .

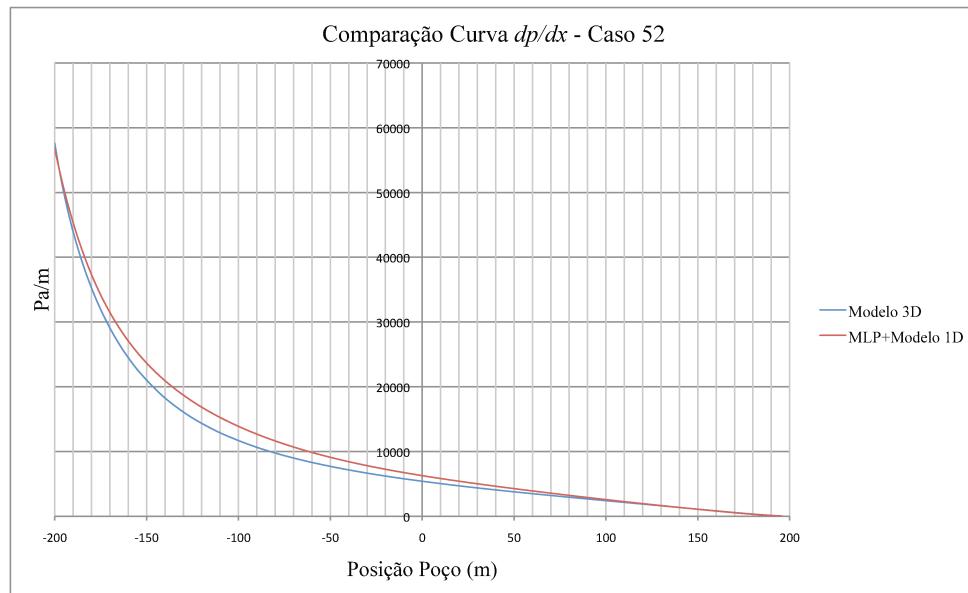


Figura 4.26: Comparação modelo 3D e *MLP+modelo 1D*, caso 52 - curva  $dp(x)/dx(x)$ .

Pode-se observar que, para o caso 100, as topologias das curvas dos dois modelos parecem estar bem próximas, o que era esperado pelos resultados do treinamento da arquitetura. Para o caso 52, porém, pode-se notar diferenças topológicas razoáveis entre as curvas, notavelmente peculiar nas curvas  $p(x)$ , entre as quais existe um  $\Delta p$  relativamente alto na região do dedão do poço. Para as curvas  $Q_{hw}(x)$  e  $dp(x)/dx$ , as diferenças ocorrem na região interna do poço. Já para a curva  $dQ_{hw}(x)/dx$ , existe ligeira diferença tanto no interior quanto nos extremos, embora a curvatura geral do modelo unidimensional esteja em conformidade com a do modelo tridimensional. Embora seja um caso mais crítico em termos do treinamento das *MPLs*, o caso 52 apresentou um erro percentual (norma  $L^1$  alterada) para vazão total  $Q_{hw}$  em torno de 2%, o que é considerado um bom resultado.

Algumas discussões e conclusões sobre os resultados do processo de treinamento e da resolução do modelo unidimensional são descritas no próximo capítulo.

# Capítulo 5

## Conclusão

O presente trabalho concentrou no estudo e desenvolvimento de redes neurais artificiais com intuito não somente da capacitação pessoal, mas também em aplicá-las num problema numérico de engenharia. Assim sendo, não somente uma biblioteca para criação de redes neurais foi desenvolvida, mas também uma metodologia para utilização de suas potencialidades como função aproximadora foi elaborada a fim de analisar possível substituição (em certas ocasiões) de um modelo tridimensional por um modelo unidimensional simplificado.

O desenvolvimento da biblioteca de redes neurais artificiais, *NeuralLib*, teve o intuito de permitir ao grupo de pesquisa, no qual o trabalho foi realizado, uma ferramenta que pudesse atender às necessidades com relação à simulação numérica e também servir como base de ensino dessa tecnologia. Além do mais, futuras implementações de forma a ampliar as potencialidades de uso e aplicação poderão ser realizadas, já que a estrutura do código como todo foi projetada para isso. A sua aplicação no presente trabalho denota uma validação um tanto quanto plausível, porém, validações futuras ainda poderão ocorrer a fim de deixá-la mais confiável e robusta. Um outro recurso adicional proveniente do fato de se desenvolver uma biblioteca é o de abrir um outro ramo de pesquisa e desenvolvimento, já que se trata de um assunto relativamente novo e promete aplicações das mais variadas.

A metodologia descrita e articulada com relação à simulação numérica do acoplamento poço-reservatório procurou aplicar as *MLPs* num problema numérico específico, utilizando-se de suas capacidades de mapear e generalizar funções a partir de conjunto de padrões de treinamento. Nesse caso, os resultados mostraram-se adequados aos objetivos do trabalho. As *MLPs* utilizadas mostraram um bom treinamento quanto aos conjuntos de dados gerados pelo modelo tridimensional, e os resultados de treinamento e generalização poderiam indicar qual das três arquiteturas seria mais adequada (no caso, *MLP-5*). No entanto, a análise nos erros das curvas geradas pelo modelo unidimensional arranjada com as *MLPs* parece indicar outra arquitetura como a mais adequada

(MLP-10, nesse caso). Essas diferenças provêm do fato de que os valores de saída das *MLPs* são coeficientes de uma função polinomial, a qual é utilizada na resolução de duas equações diferenciais acopladas. Em outros termos, cada saída da rede possui influência sobre o comportamento da função polinomial diferente uma da outra: uns parâmetros geram mais sensibilidade, outros não. Assim, um erro maior numa saída (ou parâmetro), mesmo que isso não gerasse uma energia total do erro grande para dado caso de generalização (o que indicaria um bom resultado da rede), poderia gerar um comportamento na função polinomial de forma a produzir um erro (em norma  $L^2$ , por exemplo) relativamente grande em relação a outros casos para os quais a energia total do erro foram maiores. Esse resultado são inerentes da metodologia desenvolvida, já que as saídas da rede neural não foram utilizadas como valores objetivos em si, mas sim como parâmetros de uma outra função, a qual é utilizada na resolução de equações diferenciais. Assim, a propagação e potencialização do erro ao longo desse processo todo acaba sendo algo inevitável e até, por assim dizer, curiosamente interessante.

Uma possível forma de se contornar esses pormenores seria procurar avaliar os resultados das redes neurais por uma outra medida de erro, de forma a contabilizar as variações de cada saída individualmente e não somente pela energia total. Isso poderia auxiliar a avaliação do treinamento das redes, buscando uma minimização mais equalizada nos erros dos valores de saída. Tal metodologia teria muita utilidade no presente trabalho e poderá, futuramente, ser facilmente implementada na biblioteca *NeuralLib*. Contudo, é algo que será elencado para trabalhos futuros.

Conclui-se, por fim, que os resultados, tanto do treinamento das redes neurais quanto do arranjo dessas com o modelo unidimensional do acoplamento poço-reservatório, atendem aos objetivos do trabalho. Tais resultados poderão produzir futuras aplicações não somente para indústria do petróleo, mas também para simulação numérica em problemas comuns na engenharia.

# Referências Bibliográficas

- ALRUMAH, Muhammad. **Neural Networks Predict Well Inflow Performance.** 2003. Dissertação de Mestrado. Texas A&M University, Texas, USA.
- AMINZADEH, F.; BARHEN, J.; GLOVER, C.W. e TOOMARIAN, N.B. Reservoir parameter estimation using a hybrid neural network. **Computers and Geosciences**, vol. 26, 2000.
- ARRIETA, Juan Carlos Galvis. **Finite Elements For Well-Reservoir Coupling.** 2004. Dissertação de Mestrado. Instituto Nacional de Matemática Pura e Aplicada, Rio de Janeiro, RJ.
- ARTURO, N.V.C.; SANTOS, D.V.; MENDES, J.R.P.; MIURA, K. e MOROOKA, C.K. Estudo do acoplamento poço-reservatório para poços horizontais. In **Anais do 4o. Congresso Brasileiro de P&D em Petróleo e Gás.** Campinas, SP, 2007.
- AZIZ, K. e SETTARI, A. **Petroleum Reservoir Simulation.** Appied Science Publishers, England, 1983.
- BAZARAA, M.S.; SHERALI, H.D. e SHETTY, C.M. **Nonlinear Programming - Theory and Algorithms.** John Wiley and Sons, Inc., U.S.A., 2 ed., 1993.
- BECKER, E.B.; CAREY, G.F. e ODEN, J.T. **Finite Elements: An Introduction,** vol. I. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, USA, 1981.
- BRAGA, A.; DE LEON FERREIRA DE CARVALHO, A.P. e LUDERMIR, T.B. **Redes Neurais Artificiais: Teoria e Aplicações.** LTC, Rio de Janeiro, RJ, Brasil, 2 ed., 2007.
- BURDEN, R.L. e FAIRES, J.D. **Análise Numérica.** Cengage Learning, São Paulo, SP, 8 ed., 2008.
- BUSSAB, W.O. e MORETTIN, P.A. **Estatística Básica.** Atual Editora, São Paulo, SP, 4 ed., 1997.
- CRICHLOW, H.B. **Modern Reservoir Engineering - A Simulation Approach.** Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1977.
- CUNHA, M.C.C. **Métodos Numéricos.** Editora da Unicamp, Campinas, SP, 2 ed., 2009.

- DEVLOO, P.R.B.; RYLO, E.C.; LONGHIN, G.C.; FORTI, T.L.; FARIAS, A.M.; LUCCI, P.C.A. e FERNANDES, P.D. Desenvolvimento de ferramenta computacional para cálculo de índice de produtividade de poços verticais e horizontais. In **Anais do ENAHPE**. Campos do Jordão, SP, 2009.
- DICKSTEIN, F.; LARA, A.Q.; NERI, C. e PERES, A.M. Modeling and simulation of horizontal wellbore-reservoir flow equations. In **Latin American and Caribbean Petroleum Engineering Conference**, Paper SPE 39064-MS. Society of Petroleum Engineers, Rio de Janeiro, RJ, Brasil, 1997.
- ESCOBAR, F.H. e MONTEALEGRE, M. A more accurate correlation for the productivity index fo horizontal wells. **Journal of Engineering and Applied Sciences**, vol. 3, 70–78, 2008.
- FAUSETT, L.V. **Fundamentals of Neural Networks: Architectures, Algorithms, and Applications**. Prentice-Hall, Upper Saddle River, New Jersey, USA, 1 ed., 1994.
- FERNANDES, P.D.; DA SILVA, M.G.F. e BEDRIKOVETSKY, P. Uniformização de fluxo em poços horizontais. In **Anais do ENAHPE**. Pedra Azul - Domingos Martins, ES, 2006.
- GALUSHKIN, A.I. **Neural Networks Theory**. Springer, Moskva Region, Rússia, 2007.
- GOMES, José Adilson Tenório. **Simulação Numérica de Poços Horizontais em Reservatórios com Fluxo Multifásico, Usando Refinamento Local**. 1990. Dissertação de mestrado. Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas, SP.
- HASSOUN, M.H. **Fundamentals of Artificial Neural Networks**. A Bradford book, The MIT Press, Massachusetts Institute of Technology, USA, 1 ed., 1995.
- HAYKIN, S. **Redes Neurais: Princípios e Prática**. Bookman, Porto Alegre, RS, Brasil, 2 ed., 2001.
- JOSHI, S.D. **Horizontal Well Technology**. PennWell Books, Tulsa, Oklahoma, USA, 1991.
- JUNIOR, U.S.; FERNANDES, P.D.; DE ASSIS RESSEL PEREIRA, F. e REIS, M.V.F. Estudo do acoplamento poço-reservatório: Uso de ferramentas de cfd para análise do escoamento no entorno do poço. In C. Petrobras, editor, **Boletim Técnico da Produção de Petróleo**, vol. 2, páginas 255–272. Petrobras/CENPES, Rio de Janeiro, Dezembro 2007.
- KOVÁCS, Z.L. **O Cérebro e a Sua Mente: Uma Introdução à Neurociência Computacional**. Edição Acadêmica, São Paulo, SP, Brasil, 1997.

- KOVÁCS, Z.L. **Redes Neurais Artificiais: Fundamentos e Aplicações: um texto básico.** Livraria da Física, São Paulo, SP, Brasil, 4a. ed., 2006.
- KREYSZIG, E. **Introductory Functional Analysis with Applications.** John Wiley and Sons, Inc., U.S.A., 1978.
- LEMOS, Walter Petrone. **Acoplamento Poço-Reservatório para Análise de Testes em Poços não Surgentes.** 1993. Dissertação de Mestrado. Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas, SP.
- LIPPMAN, S.B. e LAJOIE, J. **C++ Primer.** Addison-Wesley, Reading, Massachusetts, USA, 3 ed., 1998.
- LUCCI, Paulo Cesar A. **Descrição Matemática de Geometrias Curvas por Interpolação Transfinita.** 2009. Dissertação de Mestrado. Faculdade de Engenharia Civil, Arquitetura e Urbanismo, Universidade Estadual de Campinas, Campinas, SP.
- MCCULLOCH, W.S. e PITTS, W.H. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, vol. 5, 115–133, 1943.
- NGUYEN, H.H.; CHAN, C.W. e WILSON, M. Prediction of oil well production using multi-neural networks. In **IEEE 2002 Canadian Conference on Electrical and Computer Engineering, CCECE'02.** Winnipeg, Manitoba, Canada, 2002.
- PIMENTEL, W.R.O.; LISBÔA, A.C.L. e MOREIRA, D.R.R. Modelagem via redes neurais artificiais de uma unidade industrial de craqueamento catalítico fluido. In **3º Congresso Brasileiro de P&D em Petróleo e Gás.** Salvador, 2005.
- PRESS, W.H.; TEUKOLSKY, S.A.; VETTERLING, W.T. e FLANNERY, B.P. **Numerical Recipes: The Arte of Scientific Computing.** Cambridge University Press, New York, NY, 3 ed., 2007.
- ROSA, A.J.; DE SOUZA CARVALHO, R. e XAVIER, J.A.D. **Engenharia de Reservatórios de Petróleo.** Editora Interciência, PETROBRAS, Rio de Janeiro, 2006.
- SHIRDEL, M. e SEPEHRNOORI, K. Development of a coupled compositional wellbore/reservoir simulator for modeling pressure and temperature distribution in horizontal wells. In **SPE Annual Technical Conference and Exhibition**, Paper SPE 124806-MS. Society of Petroleum Engineers, New Orleans, Louisiana, USA, 2009.
- THOMAS, J.E. **Fundamentos de Engenharia de Petróleo.** Editora Interciência, PETROBRAS, Rio de Janeiro, 2001.

VICENTE, Ronaldo. **A Numerical Model Coupling Reservoir and Horizontal Well Flow Dynamics.** 2000. Phd thesis. Pennsylvania State University, Pennsylvania.

VICENTE, R.; SARICA, C. e ERTEKIN, T. Horizontal well design optimization: A study of the parameters affecting the productivity and flux distribution of a horizontal well. In **SPE Annual Technical Conference and Exhibition**, Paper SPE 84194-MS. Society of Petroleum Engineers, Denver, Colorado, USA, 2003.

VILLANUEVA, J.M.M. Modelo de aproximação de um simulador de produção de petróleo utilizando redes neurais artificiais. **Revista de Inteligência Computacional Aplicada (RICA), PUC-RJ,** , nº 1, abril 2008.

ZUBEN, Fernando José Von. **Modelos Paramétricos e Não-Paramétricos de Redes Neurais Artificiais e Aplicações.** 1996. Tese de Doutorado. Faculdade de Engenharia Elétrica e de Computação, UNICAMP.

# Apêndice A

## *NeuralLib*

A *NeuralLib* foi desenvolvida em linguagem C++. Essa linguagem foi escolhida por possuir diversas características desejáveis em simulação e análise numérica, tais como orientação a objetos, herança e polimorfismo, concepção em *templates*, entre outros tantos. Além do mais, essa é a linguagem que o grupo de pesquisa do Laboratório de Mecânica Computacional da Faculdade de Engenharia Civil, Arquitetura e Urbanismo da UNICAMP utiliza no desenvolvimento de simulações e manutenção do ambiente de programação em elementos finitos. Dessa forma, a *NeuralLib* pode ser facilmente utilizada nos programas desenvolvidos no laboratório.

Na sequência, serão descritas as classes desenvolvidas e ilustrado um caso de uso. Será exemplificado também detalhes sobre um arquivo de treinamento.

### A.1 Descrição das Classes

O uso de *templates* para criação de algumas classes foi necessário para possibilitar a extensão da biblioteca para diversos tipos de neurônios, funções de ativação, arquiteturas de rede e mesmo padrões de aprendizagem, compondo assim de fato a caracterização de uma biblioteca. Além do mais, a concepção do código permite facilidade para suportes e implementação de funcionalidades diversas, tais como análise da evolução do erro de treinamento, análise de problemas com saturação dos neurônios etc. Na sequência, descrever-se-á a lista das classes desenvolvidas e uma breve descrição de cada uma:

1. *abstLayer*. Uma classe abstrata, a qual possui todos os métodos virtuais. A ideia é conceber e formalizar uma *layer* ou camada de uma rede neural. Os métodos virtuais prescrevem a passagem de sinal para frente e para trás, ajusta os valores dos pesos sinápticos e *biases* dos neurônios, os valores de saída da rede, entre outros. Por serem métodos virtuais, a implementação fica a cargo das classes derivadas.

2. *layer<GActFun>*. Classe derivada da *abstLayer*. Essa classe *template* implementa os métodos definidos na classe mãe, de acordo com o tipo de função de ativação *GActFun*. Além dos métodos, a classe possui dois atributos importantes: um vetor de *neuron<GActFun>* e um vetor de valores do tipo *double*. O primeiro vetor, de fato, são os neurônios da camada ou *layer*, definidos também de acordo com o tipo de função de ativação. O segundo vetor corresponde aos valores de saída de cada neurônio, ou seja, saída da própria *layer*.
3. *HeavysideFunction*. Classe que implementa a função de ativação do tipo *Heavyside*. Apresenta somente dois métodos: um para processar a função e outra para processar a sua derivada. Todos os parâmetros necessários para processar são passados via método.
4. *LinearFunction*. Classe que implementa a função de ativação do tipo *Linear*. Apresenta somente dois métodos: um para processar a função e outra para processar a sua derivada. Todos os parâmetros necessários para processar são passados via método. Possui dois atributos: o valor de saída da função de ativação e o sinal de entrada. Esses atributos são usados para os cálculos internos.
5. *LogistFunction*. Classe que implementa a função de ativação do tipo *Logistic*. Apresenta somente dois métodos: um para processar a função e outra para processar a sua derivada. Todos os parâmetros necessários para processar são passados via método. Possui um atributo: o valor de saída da função de ativação, o qual é usado para os cálculos internos.
6. *TanhFunction*. Classe que implementa a função de ativação do tipo *Hyperbolic Tangent*. Apresenta somente dois métodos: um para processar a função e outra para processar a sua derivada. Todos os parâmetros necessários para processar são passados via método. Possui um atributo: o valor de saída da função de ativação, o qual é usado para os cálculos internos.
7. *SineFunction*. Classe que implementa a função de ativação do tipo *Sine*. Apresenta somente dois métodos: um para processar a função e outra para processar a sua derivada. Todos os parâmetros necessários para processar são passados via método. Possui um atributo: o valor de entrada da função de ativação, o qual é usado para os cálculos internos.
8. *network*. Classe que armazena os componentes de uma rede neural. Possui diversos atributos necessários para definição de uma rede, armazenando números de neurônios e os tipos de funções de ativação para cada *layer*, assim como os parâmetros necessários a cada função; possui um vetor que armazena todas as *layers*, um vetor dos valores de saída da própria rede, entre outros. Os métodos operam sobre cada *layer* e, consequentemente, sobre cada neurônio, acessando assim qualquer unidade ou neurônio da rede de acordo com o tipo de estrutura utilizada.

9. *neuron*<*GActFun*>. Classe *template* que implementa as funcionalidades de um neurônio artificial, de acordo com a definição da função de ativação *GActFun*. Possui diversos métodos que operam sobre as funções de ativação e sobre a inicialização dos pesos sinápticos e *bias*. Possui diversos atributos, dentre os quais destacam-se o vetor dos pesos sinápticos e o valor do *bias*, além do valor de saída do neurônio.
10. *MLPerceptron*<*GTypeTraining*>. Classe *template* que implemanta a estrutura de uma rede neural do tipo *Perceptron* de Múltiplas Camadas (*MLP*). O único atributo é um objeto do tipo *network*, que armazena todos os dados necessários da rede. Possui alguns métodos que operam sobre esse objeto, de forma a compor a estrutura de uma *MLP*, a partir do gerenciamento de fluxos de dados de uma *layer* para outra. O treinamento da rede é realizado de acordo com o tipo definido, *GTypeTraining*, passado como argumento na definição da classe.
11. *backpropagation*. Classe que implementa o algoritmo de retropropagação, utilizado no treinamento das *MLPs*. A classe possui métodos necessários para execução do algoritmo, além do próprio algoritmo. Possui também diversos atributos que armazenam dados necessários para execução do treinamento, tais como vetores das energias de erro, valores limites para número máximo de iterações e valor da energia média mínima, taxa de treinamento, entre outros. O critério de parada do algoritmo é definido nessa classe e sua estrutura permite a implementação de *logs* para verificação do desempenho do treinamento ao longo das iterações.
12. *networkdata*. Classe que armazena, estrutura e gerencia os dados que entram e saem numa rede neural. É a classe cujo objeto faz a interface entre o usuário e a rede. Possui diversos métodos, todos dedicados ao gerenciamento dos dados de treinamento, escalonamento dos dados, leitura e salvamento dos arquivos correspondentes aos casos de treinamento etc. Possui vários atributos correspondentes aos diversos conjuntos de treinamento (validação, teste etc), assim como valores escalonados, entre outros. O objeto dessa classe é passado como argumento para os métodos de treinamento e generalização da classe de arquitetura, por exemplo, *MLPerceptron*<*GTypeTraining*>. A classe *networkdata* pode receber diversas novas implementações referentes ao tratamento e salvamento dos dados, de acordo com a necessidade de acoplar as redes neurais em outros ambientes ou programas.

## A.2 Caso de Uso

Para ilustrar a utilização da *NeuralLib* na geração de redes do tipo *Multilayer Perceptron* no mapeamento de funções, é apresentado uma sequência de definições de objetos e parâmetros específicos para um caso de uso da *MLP*. As etapas em destaque apresentam e definem as variáveis necessárias e estão escritas em linguagem C++.

### Definição dos Arquivos

Os arquivos a serem definidos são: arquivo de dados de treinamento, no qual estão os valores de *input* e *output* dos padrões a serem mapeados; arquivo de resultado, no qual são escritos os valores de *input* originais e os de *output* produzidos pela rede; arquivo da rede treinada, no qual são escritos a configuração da rede e os valores dos pesos sinápticos e *biases* após o treinamento.

```
string TrainFile = "C:\RNA.txt";
string TrainFileOut = "C:\RNA_Out.txt";
string TrainedNetworkFile = "C:\TrainedNetwork.txt";
ifstream TrainData( TrainFile.c_str() );
ofstream TrainDataOut( TrainFileOut .c_str() );
ofstream TrainedNetworkData( TrainedNetworkFile.c_str() );
```

### Parâmetros Gerais

Os parâmetros gerais são: tratamento dos dados de *input* e *output*, o qual escalona todos os valores para intervalos específicos; número de camadas e de neurônios em cada uma; tipos de neurônios em cada camada; parâmetros do processo de treinamento.

### Tratamento Dados de *Input* e *Output*

```
double MaxLimitIn = 1.0;
double MinLimitIn = -1.0;
double MaxLimitOut = 1.0;
double MinLimitOut = -1.0;
```

### Número de Camadas e Neurônios

```
int InputNum = 8;
int OutputNum = 7;
int LayerNum = 3;
std::vector<int> NeuronForLayer( LayerNum );
```

```

NeuronForLayer[0] = InputNum;
NeuronForLayer[1] = 10;
NeuronForLayer[2] = OutputNum;
int TotalNeuronNum = InputNum + OutputNum + NeuronForLayer[1];

```

### Tipos de Neurônios nas Camadas

```

std::vector<int> ActFuncType( LayerNum );
std::vector<double> a( LayerNum );
std::vector<double> b( LayerNum );
ActFuncType[0] = 2;
ActFuncType[1] = 2;
ActFuncType[2] = 2;
a[0] = 1.7159;
a[1] = 1.7159;
a[2] = 1.7159;
b[0] = 2.0/3.0;
b[1] = 2.0/3.0;
b[2] = 2.0/3.0;

```

### Processo de Treinamento - *backpropagation*

```

double n = 0.05;
double alpha = 0.0000001;
double MaxValueError = 0.001;
int ErrorCriteriaType = 2;
int MaxIteratorNumber = 10000;
int K0 = 500;

```

### Definição da Rede

Os parâmetros da rede neural são armazenados em uma estrutura de dados, a qual os concentra e os disponibiliza para outros objetos.

```

network MyNetwork( LayerNum, NeuronNum, NeuronForLayer );
MyNetwork.SetTypeFunctionParam( ActFuncType, a, b );

```

## Definição da Arquitetura

A arquitetura da rede a ser utilizada é definida com base na estrutura de dados *network*. Define-se também o tipo de treinamento da arquitetura.

```
MLPerceptron<backpropagation> MyMLP;  
MyMLP.DefineNetwork( MyNetwork );
```

## Definição do Treinamento

Os parâmetros de treinamento da arquitetura são definidos. Cada tipo de treinamento possui parâmetros específicos. No exemplo, são definidos os que são necessários para o algoritmo *backpropagation*.

```
backpropagation MyTraining;  
MyTraining.SetTypeCritStop( ErrorCriteriaType, MaxValueError );  
MyTraining.SetN( n );  
MyTraining.SetAlpha( alpha );  
MyTraining.SetK0( K0 );  
MyTraining.SetNumberMaxIteration( MaxIteratorNumber );
```

## Definição do Network Data

A estrutura de dados que armazena todos os dados de entrada e saída da rede neural é definida. Essa estrutura faz um tratamento dos padrões antes do processo de treinamento e gerencia todo fluxo de dados que entra e sai da rede neural.

```
networkdata MyData;  
MyData.SetSchedullingParam( MaxLimitIn, MinLimitIn, MaxLimitOut, MinLimitOut );  
MyData.LoadData( TrainData ) ;
```

## Treinamento e Generalização

Essa etapa é o acionamento do processo de treinamento utilizando os conjuntos de treinamento e de validação. Pode-se também realizar o processo de generalização, o qual executa todo os casos presentes no arquivo de treinamento, ou seja, os de treinamento, os de validação e os de teste.

```
MyMLP.TrainNetwork( MyTraining, MyData );  
MyMLP.RunNetwork( MyData );
```

### Salvamento dos Resultados

Após o processo de treinamento e generalização, faz-se o salvamento dos resultados e da rede treinada. O arquivo de resultados é estruturado semelhantemente ao de treinamento, porém os valores de *output* são os produzidos pela rede neural. O arquivo da rede treinada apresenta todos os parâmetros necessários para reconstruí-la posteriormente.

```
MyData.SaveAllResults( TrainDataOut );  
MyMLP.SaveTrainedNetwork( TrainedNetworkData );
```

### A.3 Arquivo Treinamento

O arquivo de treinamento contem todos os padrões que são utilizados no processo de treinamento da rede. Os padrões são agrupados em subconjuntos específicos: treinamento, validação e teste. O preenchimento do arquivo deve seguir essa sequência. Apenas para ilustrar, a Tabela A.1 apresenta um trecho de arquivo de treinamento, no qual são apresentados os conjuntos de treinamento, de validação e de teste.

9							
7							
50							
25							
25							
0	1233	833	7	(...)	-1.03684	0.284343	(...)
1	3314	2914	7	(...)	-2.29589	0.409811	(...)
2	2223	1823	11	(...)	-1.21972	0.774174	(...)
				:			
0	1853	1453	8	(...)	-1.21977	0.466505	(...)
1	3032	2632	5	(...)	-1.94562	1.26945	(...)
2	3353	2953	17	(...)	-1.81301	1.14936	(...)
				:			
0	3123	2723	8	(...)	-2.16192	0.367926	(...)
1	2300	1900	11	(...)	-1.24067	0.986081	(...)
2	1892	1492	6	(...)	-1.57812	0.104688	(...)
				:			

Tabela A.1: Trecho de um arquivo de treinamento de uma rede neural.

Os primeiros dois números do arquivo referem-se ao número de *input* e *output*, respectivamente. Os três números seguintes são as quantidades de padrões existentes nos subconjuntos de treinamento, validação e de teste, respectivamente. Os números na sequência são os padrões dos subconjuntos. A primeira coluna é o número ou *ID* do padrão. A sequência de números após o *ID* são os valores de *input*, seguidos pelos valores de *output*. Ressalta-se que os *IDs* são numerações específicas de cada subconjunto.

Embora seja denominado arquivo de treinamento, esse arquivo pode ser usado apenas para testar casos em uma rede já treinada. Nessa situação, define-se como 0 (zero) as quantidades de casos para os subconjuntos de treinamento e validação, preenchendo apenas o de teste.