Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia
COPPE UFRJ

METODOLOGIAS DE ANÁLISE, SÍNTESE E OTIMIZAÇÃO DE SISTEMAS PARA PRODUÇÃO DE PETRÓLEO OFFSHORE ATRAVÉS DE METAMODELOS E ENXAME DE PARTÍCULAS

Aline Aparecida de Pina

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Civil.

Orientador(es): Breno Pinheiro Jacob
Beatriz de Souza Leite Pires de Lima

Rio de Janeiro
Dezembro de 2010

METODOLOGIAS DE ANÁLISE, SÍNTESE E OTIMIZAÇÃO DE SISTEMAS PARA PRODUÇÃO DE PETRÓLEO OFFSHORE ATRAVÉS DE METAMODELOS E ENXAME DE PARTÍCULAS

Aline Aparecida de Pina

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA CIVIL.

Examinada por:

Prof. Breno Pinheiro Jacob, D.Sc.

Profª. Beatriz de Souza Leite Pires de Lima, D.Sc

Prof. Nelson Francisco Favilla Ebecken, D.Sc

Prof. Hélio José Corrêa Barbosa, D.Sc

Prof. Celso Kazuyuki Morooka, D.Sc

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 2010

Pina, Aline Aparecida de

Metodologias de Análise, Síntese e Otimização de Sistemas para Produção de Petróleo Offshore Através de Metamodelos e Enxame de Partículas / Aline Aparecida de Pina. – Rio de Janeiro: UFRJ/COPPE, 2010.

VIII, 176 p.: il.; 29,7 cm.

Orientadores: Breno Pinheiro Jacob

Beatriz de Souza Leite Pires de Lima

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Civil, 2010.

Referências Bibliográficas: p. 165-176.

1. Otimização de Sistemas Offshore. 2. Metamodelos. 3. Enxame de Partículas. I. Jacob, Breno Pinheiro *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

AGRADECIMENTOS

Agradeço a todos aqueles que direta ou indiretamente tenham colaborado de alguma forma para a realização deste trabalho.

Agradeço ao apoio financeiro fornecido pela CAPES durante o decorrer do trabalho. Enquanto existirem pessoas interessadas em financiar pesquisas para o avanço da tecnologia e consequente melhoria da sociedade, o homem continuará evoluindo.

Agradeço aos professores Breno Pinheiro Jacob e Beatriz de Souza Leite Pires de Lima por terem aceitado ser meus orientadores, pela confiança, pela paciência, pela compreensão, por toda ajuda e atenção que me deram durante o desenvolvimento deste trabalho.

Agradeço ao professor Carl Horst Albrecht pela grande ajuda durante a minha procura por um novo curso de Doutorado e pelo fundamental auxílio no desenvolvimento deste trabalho.

Agradeço ao professor Cláudio Esperança por sempre estar disposto a me ajudar quando preciso, desde o meu curso de mestrado. Sua ajuda foi simples, mas fundamental para que eu pudesse continuar meus estudos na nova área que escolhi.

Agradeço aos meus colegas do LAMCSO (Laboratório de Métodos Computacionais e Sistemas *Offshore*), em especial, Bruno Monteiro, Mauro Henrique e Ivete que me ajudaram diretamente em alguma etapa de desenvolvimento desta pesquisa.

Agradeço à todos os funcionários da secretaria do Programa de Engenharia Civil da COPPE/UFRJ que resolveram tão pacientemente os meus problemas burocráticos, em especial Elizabeth Cornelo, Jairo Leite e Rita Lisboa.

Agradeço à minha Família pelo apoio incondicional demonstrado em todos os momentos. É muito bom saber que existe alguém sempre torcendo e vibrando com nossas conquistas, rezando para que tudo transcorra sem problemas. Essa força foi e sempre será essencial para o término deste e de outros futuros trabalhos.

Acima de tudo e de todos, agradeço a DEUS pela chance de poder desenvolver este trabalho, completando mais um ciclo de minha vida profissional, sempre com muito esforço, mas com a certeza de ter realizado um bom trabalho. Muito obrigada.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

METODOLOGIAS DE ANÁLISE, SÍNTESE E OTIMIZAÇÃO DE SISTEMAS PARA PRODUÇÃO DE PETRÓLEO OFFSHORE ATRAVÉS DE METAMODELOS E ENXAME DE PARTÍCULAS

Aline Aparecida de Pina

Dezembro/2010

Orientadores: Breno Pinheiro Jacob
Beatriz de Souza Leite Pires de Lima

Programa: Engenharia Civil

Este trabalho apresenta estudos sobre três metodologias de otimização capazes de facilitar o desenvolvimento de projetos de estruturas *offshore*. Para construir tais metodologias, foram empregadas duas ferramentas computacionais: o método de otimização por Enxame de Partículas (PSO) e Redes Neurais Artificiais (RNA). A primeira metodologia é baseada no uso do PSO como um método de otimização de projeto de estruturas *offshore* a fim de facilitar a busca por melhores soluções. A segunda metodologia envolve estudos sobre o uso de metamodelos para substituir a análise dinâmica de *risers* e linhas de ancoragem, a fim de reduzir o tempo computacional necessário para a obtenção de uma série temporal de tração no topo. A terceira metodologia envolve estudos sobre o uso de metamodelos de redes neurais, ajustados a partir de exemplos gerados no processo de otimização, a fim de ser capazes de aproximar diretamente a resposta dos sistemas. Através do uso dessas três metodologias, espera-se conseguir resultados para configurações de linhas de ancoragem e *risers* tão boas quanto aquelas conseguidas por meio da análise dinâmica com modelos de elementos finitos, em um tempo consideravelmente menor.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

METHODOLOGIES OF ANALYSIS, SYNTHESIS AND OPTIMIZATION OF SYSTEMS FOR OFFSHORE OIL PRODUCTION BY MEANS OF METAMODELS AND PARTICLE SWARM

Aline Aparecida de Pina

December/2010

Advisors: Breno Pinheiro Jacob
Beatriz de Souza Leite Pires de Lima

Department: Civil Engineering

This work presents studies on three optimization methodologies capable of facilitating the development of offshore structures design. In order to construct such methodologies, two computational tools were used: the Particle Swarm Optimization method (PSO) and Artificial Neural Networks (ANN). The first methodology is based on the use of PSO as an optimization method for the design of offshore structures so as to facilitate the search for better solutions. The second methodology involves studies on the use of metamodels to replace the dynamic analysis of *risers* and mooring lines, in order to reduce the computational time necessary for achieving top tension time-series. The third methodology involves studies on the use of neural networks metamodels, adjusted with examples generated in the optimization process, so as to be capable of directly approximating the systems response. By using these three methodologies, it is expect to achieve results for the configurations of mooring lines and *risers* as good as those achieved by means of dynamic analysis with finite element models, in a considerably lower time.

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto e Motivação

O desenvolvimento de atividades de exploração de petróleo *offshore* em águas profundas e ultra-profundas tem aumentado continuamente nos últimos anos. Esta tarefa envolve o uso de sistemas flutuantes de produção (Floating Production Systems - FPS), mantidos em posição por linhas de ancoragem, e conectados aos equipamentos submarinos (tais como manifolds e wellheads) pelos *risers* de produção que transportam a produção de petróleo. Muitos aspectos são envolvidos no projeto de tais sistemas, relativos principalmente à eficiência, à segurança e à redução de custos.

Um aspecto crucial de tais atividades é o projeto dos *risers* de produção (JACOB *et al.*, 1999). Diferentes tipos de *risers* são usados em sistemas flutuantes de produção, geralmente classificados como "flexíveis" ou "rígidos". Risers flexíveis são estruturas complexas, manufaturadas usando camadas com tipos diferentes de materiais, e tem sido empregados com sucesso em profundidades de água abaixo de 1000m. Para águas mais profundas, entretanto, podem alcançar limites técnicos e econômicos de praticabilidade devido à limitação do diâmetro viável do *riser*, particularmente quando associado às elevadas pressões externas, e aos significativos offsets estáticos e movimentos de heave (movimentos verticais) da plataforma.

Ultimamente, os *risers* rígidos (mais especificamente, o *riser* de aço em catenária - Steel Catenary Riser - SCR) tem se mostrado capazes de superar tais limitações. Tais *risers* são estruturas mais simples, manufaturadas de tubulações de aço. Estudos precedentes relativos ao uso dos SCR instalados em FPSs (JACOB *et al.*, 1998, 1999, 2001) consideraram diversas configurações alternativas diferentes para os SCR, incluindo a catenária livre e outras configurações com elementos de dobramento e/ou de flutuação tais como as configurações *lazy-wave*, *steep-wave*, *lazy-S* e *steep-S*. O comportamento destas configurações foi simulado por diversas análises dinâmicas não-lineares no domínio do tempo. Estes estudos mostraram, como esperado, que as configurações com elementos de flutuação apresentam um comportamento estrutural mais favorável do que a catenária livre sob carregamentos ambientais de onda, de correnteza, e dos movimentos impostos pela plataforma, às circunstâncias extremas e às condições operacionais e de fadiga.

Portanto, embora a catenária livre apresente custos menores e a praticabilidade possa ser provada em algumas situações particulares, dos resultados apresentados (JACOB *et al.*, 1999) a configuração *lazy-wave* foi selecionada como um caso base para executar estudos mais detalhados. Esta configuração pode ser definida como um arranjo em dupla catenária, com flutuadores distribuídos instalados em uma seção intermediária do *riser*. Este segmento alivia o peso suportado pela unidade flutuante, e contribui com os momentos de restauração quando submetido à cargas laterais.

Os estudos paramétricos exaustivos apresentados em (JACOB *et al.*, 1999) compreenderam uma tarefa árdua, envolvendo a variação "manual" de diferentes parâmetros geométricos e dados ambientais, a fim de conseguir uma melhor avaliação do comportamento estrutural, e determinar valores para alguns parâmetros que definem a configuração do *riser* (incluindo por exemplo o ângulo de topo, o comprimento e a posição da seção com flutuadores). O objetivo era encontrar uma configuração com custos razoáveis e, ainda assim, capaz de resistir a carregamentos extremos e de fadiga.

Entretanto, tais estudos foram executados baseados em um cenário particular, e as conclusões não podem ser prontamente prolongadas para outros cenários, por exemplo em águas ultra-profundas, onde estudos paramétricos similares grandes e custosos seriam necessários.

Outro aspecto importante no projeto de sistemas flutuantes de produção é o projeto de sistemas de ancoragem, que fornecem as forças de restauração que mantém a embarcação em posição. Podem ser colocadas em catenária (em sistemas de ancoragem convencionais), em configuração taut-leg (pernas retesadas), ou verticalmente em forma de tendão. Os materiais mais usados em linhas de ancoragem são correntes de aço, cabos de aço e mais recentemente cabos do poliéster (ROSSI *et al.*, 2001). Geralmente, as correntes são usadas nas partes iniciais e finais das linhas de ancoragem, porque são mais resistentes à manipulação e à fricção com o solo marinho e com os molinetes das plataformas.

O sistema de ancoragem deve ser capaz de fornecer uma rigidez de forma que o movimento da unidade seja mínimo, sem que as forças envolvidas excedam limites de segurança pré-estabelecidos (API, 1996). A escolha correta da topologia da ancoragem e da configuração de cada linha é crucial para o desempenho do sistema flutuante, como pode ser visto no sistema DICAS (GARZA-RIOS *et al.*, 1997; MASETTI, 1997).

Para a análise de projeto de sistemas de ancoragem para FPSs, a tendência atual considera o uso de ferramentas de análise dinâmica não-linear, consistindo em

estratégias de análise desacopladas e/ou acopladas. Na estratégia desacoplada, os modelos de elementos finitos de cada linha são analisados individualmente, submetidos aos movimentos previamente calculados da plataforma. Por outro lado, na estratégia de análise acoplada, um modelo hidrodinâmico de um sistema flutuante é acoplado a um modelo de elementos finitos estrutural e hidrodinâmico das linhas de ancoragem, a fim calcular simultaneamente os movimentos da plataforma e a resposta estrutural das linhas. Em ambos os casos, ferramentas para a simulação dinâmica no domínio do tempo devem ser aplicadas e um número elevado de análises deve ser executado, exigindo o tempo computacional elevado.

Portanto, a seleção de configurações de sistemas de ancoragem e de *risers* com bom desempenho estrutural e baixo custo deve ser formalmente descrita e tratada como um problema de otimização. Este fato já havia sido reconhecido em trabalhos anteriores (VIEIRA *et al.*, 2003; de LIMA *et al.*, 2005; VIEIRA, 2008; VIEIRA *et al.*, 2008a,b; VIEIRA, 2009), onde foram empregados diversos tipos de métodos evolutivos no desenvolvimento de procedimentos de otimização para *risers lazy-wave*. Em (RODRIGUES, 2004) foi empregado algoritmos genéticos na otimização de sistemas híbridos de *risers* baseados no conceito de bóia de subsuperfície. Além disso, em (ALBRECHT, 2005; MONTEIRO, 2008) foram aplicados micro algoritmos genéticos e enxame de partículas no desenvolvimento de procedimentos de otimização para linhas de ancoragem.

Embora o uso de métodos de otimização torne o projeto desses sistemas mais fácil, evitando uma análise paramétrica manual exaustiva, a análise dinâmica do modelo de elementos finitos ainda deve ser aplicada em cada etapa do algoritmo de otimização.

Os resultados de uma análise no domínio do tempo consistem na série temporal dos parâmetros de resposta de interesse, como: esforços, tensões, movimentos, etc. É importante ressaltar que, a fim conseguir estabilidade estatística nos resultados obtidos na análise extrema, assim como a análise de fadiga, a série temporal deve ser suficientemente longa, o que faz com que cada um das análises necessárias exijam um tempo computacional elevado.

Geralmente, quando há dificuldades em estabelecer ou aplicar um modelo matemático para investigar um problema físico, modelos substitutos mais simples são usados para descrever a resposta dinâmica do sistema (CASSINI & AGUIRRE, 1999), o que permite a predição da resposta de sistemas dinâmicos em função somente dos dados

de entrada, não exigindo o conhecimento prévio de parâmetros dinâmicos do sistema (PINA & ZAVERUCHA, 2008).

Alguns modelos com estas características foram usados em problemas práticos de engenharia. Redes neural artificiais foram aplicadas em diversos problemas de otimização, substituindo computacionalmente cálculos custosos (MAZAHERI & DOWNIE, 2004; MAHFOUZ, 2006; HESS *et al.*, 2006; BAZARGAN *et al.*, 2006). KITAMURA *et al.* (2001) aplica algoritmos genéticos na otimização de navios de carga e substitui a análise de elementos finitos por redes neurais.

Na análise dinâmica de estruturas *offshore*, GUARIZE (2004) e MATOS (2005) verificaram a possibilidade de aplicação de redes neurais artificiais, obtendo bons resultados. Em (GUARIZE *et al.*, 2007), foi apresentado um procedimento híbrido eficiente, usando redes neurais e elementos finitos, 20 vezes mais rápido do que uma simulação completa usando somente elementos finitos.

Sendo assim, o uso de metamodelos na predição de séries temporais de análises dinâmicas mostrou ser uma boa ferramenta para diminuir o custo computacional de procedimentos de otimização na busca por configurações ótimas de sistemas de *risers*. Para a criação de tais metamodelos basta tomar como parâmetro de entrada da rede neural os dados de movimentos da embarcação durante todo o tempo de simulação, reduzindo o tempo de análise necessário para obter os parâmetros de saída (esforços, tensões, etc). Esse procedimento deve ser repetido a cada passo do processo de otimização para cada configuração candidata.

A fim de reduzir ainda mais o custo computacional e viabilizar o uso de metamodelos em sistemas de ancoragem, onde os movimentos não são dados conhecidos (e sim resultados obtidos de análises de modelos acoplados do casco, linhas de ancoragem e dos *risers*), pode-se considerar o emprego dos metamodelos de redes neurais diretamente no processo de otimização. Técnicas de otimização baseadas em metamodelos já foram utilizadas na solução de vários problemas em diversas áreas (GREFENSTETTE & FITZPATRICK, 1985; BERNARDO *et al.*, 1992; MYERS & MONTGOMERY, 1995; DENNIS & TORCZON, 1996; OSIO & AMON, 1996; WANG *et al.*, 2001, WANG & SHAN, 2007; GOMES, 2007; FONSECA, 2009; GASPAR-CUNHA & VIEIRA, 2009). Nessas técnicas, os metamodelos são ajustados utilizando as informações obtidas durante o processo de otimização e, depois de serem devidamente ajustados, substituem a análise rigorosa durante o restante do processo. Na área de sistemas *offshore*, MENDONÇA (2004) aplicou uma estratégia híbrida paralela

utilizando redes neurais treinadas com indivíduos das populações de algoritmos genéticos (AG) na otimização de movimentos no casco de uma plataforma semi-submersível. SONG *et al.* (2010) aplicaram metamodelos de superfície de resposta de mínimos quadrados móveis na otimização de projeto baseado em segurança de um suporte de *riser* num FPSO.

A motivação deste trabalho, portanto, é analisar métodos que permitam a otimização eficiente de sistemas de ancoragem e de *risers* visando um bom desempenho estrutural de tais sistemas bem como a redução de custos.

Este tema tem atualmente grande importância prática para a indústria de petróleo, uma vez que irá permitir a definição de configurações mais eficientes e econômicas para sistemas de de ancoragem e *risers*, levando a projetos mais eficientes para tais sistemas estruturais que são fundamentais na exploração de petróleo *offshore*.

1.2 Objetivo

Este trabalho tem como objetivo o desenvolvimento de ferramentas computacionais para o projeto de sistemas para produção de petróleo *offshore*, empregando recursos de inteligência computacional para complementar e/ou substituir métodos numéricos.

1.3 Metodologia

Este trabalho apresenta estudos sobre três metodologias de otimização capazes de facilitar o desenvolvimento de projetos de estruturas *offshore* em diferentes etapas do processo.

A primeira metodologia é baseada no uso de métodos evolutivos na otimização de estruturas *offshore* e envolve estudos sobre a aplicação de uma estratégia de otimização promissora baseada em conceitos evolucionários ao projeto de *risers*: o método de Otimização por Enxame de Partículas (Particle Swarm Optimization - PSO).

O método PSO é um membro da larga categoria de métodos de Inteligência de Enxames (Swarm Intelligence) (KENNEDY *et al.*, 2001) para resolver problemas de otimização, e foi relacionado à computação evolucionária como Estratégias Evolutivas e Algoritmos Genéticos (PARSOPOULOS & VRAHATIS, 2002). Foi originalmente proposto como uma simulação do comportamento social, e inicialmente introduzido como um método de otimização em (KENNEDY & EBERHARDT, 1995a; 1995b).

Desde então este método tem atraído muita atenção nas comunidades de informática e tecnologia.

Apesar de ser relativamente recente, o método de PSO foi usado para a solução de uma variedade de problemas difíceis de otimização, apresentando em alguns casos uma taxa de convergência mais rápida do que outros algoritmos evolucionários (HU *et al.*, 2003; HASSAN *et al.*, 2005). Alguns exemplos de uso do PSO em aplicações diversas podem ser encontrados em (PENG *et al.*, 2000; ABIDO, 2002; OURIQUE *et al.*, 2002; KANNAN *et al.*, 2004; PARSOPOULOS & VRAHATIS, 2004; EBERHARDT & SHI, 2004; MEDEIROS, 2005).

Além de sua eficiência, uma outra razão para a popularidade crescente do método de PSO é que precisam ser ajustados menos parâmetros do que na maioria dos outros algoritmos evolucionários (EA), o que o torna particularmente fácil de implementar (HE *et al.*, 2004). Entretanto, como em todo EA, seu desempenho dependente dos valores selecionados para os parâmetros. Uma má escolha dos valores pode conduzir à convergência prematura a um ótimo local, ou causar um comportamento oscilatório com convergência lenta próximo à solução ótima. Algumas diretrizes básicas para a seleção de valores de parâmetros são apresentadas em TRELEA (2003), junto com uma análise teórica do algoritmo de PSO. Em todo caso, o comportamento do método é certamente dependente do problema, e análises paramétricas devem ser executadas para ajustá-lo para cada aplicação particular do algoritmo de PSO.

No que diz respeito a convergência, o comportamento do algoritmo de otimização é particularmente importante para a aplicação considerada neste trabalho - *risers* para a produção de petróleo *offshore*. Sabe-se que a avaliação do comportamento estrutural de configurações de *risers* exige análises estruturais com elementos finitos que empregam um solucionador dinâmico não-linear no domínio do tempo, que é extensivamente demorado. Logo, uma abordagem ideal de otimização deveria poder encontrar uma solução ótima com menos análises para o cálculo da fitness de cada configuração candidata, assim minimizando o esforço computacional.

Portanto, tem-se como o objetivo estudar algumas variações propostas da formulação padrão do método de PSO, e realizar uma análise da convergência do algoritmo através de muitos experimentos, testando diferentes ajustes de parâmetros. O método foi implementado em uma ferramenta computacional destinada a projeto de diferentes sistemas *offshore* que incluem *risers* e linhas de ancoragem para sistemas de produção flutuantes, a ferramenta *ProgOtim*. Esta ferramenta incorpora outros métodos

de otimização baseados em conceitos evolucionários apresentados em trabalhos anteriores (VIEIRA *et al.*, 2003; de LIMA *et al.*, 2005; ALBRECHT, 2005; VIEIRA, 2008, MONTEIRO, 2008; VIEIRA *et al.*, 2008a,b; VIEIRA, 2009; PINA *et al.*, 2008, 2010a), incluindo algoritmos genéticos, micro algoritmos genéticos, PSO e estratégias evolutivas, aplicadas ao projeto de diferentes sistemas *offshore* que incluem *risers* e linhas de ancoragem para sistemas de produção flutuantes.

Embora o uso de métodos de otimização torne mais fácil o projeto de sistemas *offshore*, evitando uma análise paramétrica manual exaustiva, a análise dinâmica do modelo de elementos finitos ainda deve ser aplicada em cada etapa do algoritmo de otimização. Por esse motivo, a segunda metodologia aqui apresentada envolve estudos sobre a aplicação de modelos substitutos (metamodelos) associados à análise dinâmica de linhas de ancoragem (sendo facilmente extensível à análise de *risers*, no entanto, optou-se por abordar outro problema essencial para o projeto de sistemas flutuantes de produção *offshore*).

Os resultados de uma análise dinâmica no domínio do tempo consistem na série temporal dos parâmetros de resposta de interesse. É importante ressaltar que, para alguns projetos de engenharia, pode ser necessária a obtenção de uma série temporal suficientemente longa, o que faz com que cada uma das análises necessárias exijam um tempo computacional elevado. Este é o caso da análise dinâmica de estruturas *offshore*, que necessita que a série temporal dos parâmetros de resposta (esforços, tensões, movimentos, etc) seja suficientemente longa, a fim de conseguir estabilidade estatística nos resultados obtidos na análise extrema, assim como na análise de fadiga.

Geralmente, quando há dificuldades em estabelecer ou aplicar um modelo matemático para investigar um problema físico, modelos substitutos (metamodelos) mais simples são usados para descrever a resposta dinâmica do sistema (CASSINI & AGUIRRE, 1999), o que permite a predição da resposta de sistemas dinâmicos em função somente dos dados de entrada, não exigindo o conhecimento prévio de parâmetros dinâmicos do sistema (PINA & ZAVERUCHA, 2008).

As Redes Neurais Artificiais (RNA) ganharam especial atenção na previsão de séries temporais devido à sua habilidade de aprendizado e sua capacidade de generalização, associação e busca paralela. Estas qualidades as tornam capazes de identificar e assimilar as características mais marcantes das séries, tais como sazonalidade, periodicidade, tendências, entre outras, na maioria das vezes camufladas

por ruídos, sem necessitar do trabalhoso passo de formulação teórica, imprescindível em procedimentos estatísticos (ABELÉM, 1994).

Além disso, as RNAs também têm se destacado pelos seguintes aspectos: sua capacidade em lidar com dados não lineares; sua robustez, pois são capazes de se auto corrigir mesmo depois de previsões erradas e; sua facilidade de utilização. É interessante ressaltar que, apesar de oferecer vantagens, as RNAs também possuem problemas. Um dos principais é a falta de procedimentos para definir com precisão o número de camadas intermediárias ou o número de neurônios em cada uma destas camadas, ou seja, a configuração mais apropriada para a aplicação estudada.

Na análise dinâmica de estruturas *offshore*, GUARIZE (2004) e MATOS (2005) verificaram a possibilidade de aplicação de redes neurais artificiais, obtendo bons resultados. Em (GUARIZE *et al.*, 2007), foi apresentado um procedimento híbrido eficiente, usando redes neurais e elementos finitos, 20 vezes mais rápido do que uma simulação completa usando somente elementos finitos. Tais análises consideraram a utilização de redes neurais como modelos com entradas exógenas, ou seja, modelos onde o valor atual da série temporal de resposta é relacionado apenas aos valores atuais e passados das séries de entrada. As entradas exógenas são as séries que ajudam a prever a variável de interesse.

Sendo assim, propõe-se, no presente trabalho, a utilização de redes neurais como modelos exógenos autoregressivos não-lineares (nonlinear autoregressive exogenous - NARX), isto é, modelos autoregressivos não-lineares que tenham entradas exógenas. Isto significa que os modelos relacionam o valor atual da série temporal a valores passados da mesma série, valores atuais e passados das séries exógenas e um termo residual. A fim de verificar a qualidade das soluções obtidas pelo modelo proposto, será feita uma comparação entre os dois modelos citados: o modelo só com entradas exógenas e o modelo NARX.

A implementação de redes neurais artificiais utilizada foi desenvolvida em MATLAB e as séries temporais testadas foram obtidas a partir da simulação de um sistema flutuante de produção composto por 8 linhas de ancoragem e 5 *risers*, numa profundidade de 2000m, usando carregamentos ambientais como entradas para o programa SITUA/PROSIM, desenvolvido no LAMCSO (Laboratório de Métodos Computacionais e Sistemas Offshore), PEC/COPPE/UFRJ.

Embora o uso de metamodelos na predição de séries temporais de análises dinâmicas possa constituir uma boa ferramenta para diminuir o custo computacional de

procedimentos de otimização na busca por configurações ótimas, esse procedimento deve ser repetido a cada passo do processo de otimização para cada configuração candidata.

A fim de reduzir ainda mais o custo computacional e viabilizar o uso de metamodelos em sistemas de ancoragem, onde os movimentos não são dados conhecidos (e sim resultados obtidos de análises de modelos acoplados do casco, linhas de ancoragem e dos *risers*), a terceira metodologia proposta envolve estudos sobre a utilização de uma estratégia de otimização de projeto baseada em metamodelagem. Nesse caso, os metamodelos de redes neurais serão empregados diretamente no processo de otimização, utilizando-se como parâmetros de entrada e saída da rede os mesmos parâmetros do algoritmo de otimização. Tomando por base a estratégia de metamodelagem com aproximação sequencial, será necessário utilizar a análise rigorosa apenas nos primeiros passos do processo de otimização a fim de construir um conjunto de dados para treinamento e validação do metamodelo. Nos passos seguintes da otimização, com a rede neural devidamente treinada, todas as configurações candidatas serão avaliadas diretamente através do metamodelo, sem ser necessário o conhecimento do comportamento estrutural inicial dos sistemas.

Dessa forma, será apresentado um estudo para verificar a viabilidade da utilização dessa estratégia de metamodelagem na otimização de projeto de um sistema de ancoragem composto por 8 linhas de ancoragem e 5 *risers*, numa profundidade de 2400m. Uma vez que o problema de otimização de sistemas de ancoragem retorna três valores de resposta que definem a qualidade da configuração (offset máximo, tração no topo máxima e fitness), propõe-se a utilização de três modelos de arquitetura para as redes neurais utilizadas, considerando-se combinações diferentes dos parâmetros de resposta da otimização como dados de saída para a rede. A fim de analisar qual dos três modelos propostos obteve o melhor desempenho, será apresentada uma análise comparativa dos resultados obtidos.

Através do uso dessas três metodologias descritas anteriormente, espera-se conseguir resultados para configurações de linhas de ancoragem e de *risers* tão boas quanto aquelas conseguidas por meio da análise dinâmica com modelos de elementos finitos, em um tempo consideravelmente menor.

1.4 Organização do Trabalho

Os Capítulos do trabalho foram organizados da seguinte forma: no Capítulo 2, são apresentadas informações sobre os sistemas mais comuns de exploração de petróleo *offshore* e as metodologias de análise frequentemente utilizadas para projeto desses sistemas. No Capítulo 3, são apresentados alguns método de otimização baseados em conceitos de evolução de população, dando maior ênfase ao algoritmo de Enxame de Partículas (PSO). No Capítulo 4, são apresentadas informações sobre metamodelagem e algumas das principais técnicas utilizadas. No Capítulo 5, são apresentados conceitos básicos de Redes Neurais Artificiais e algoritmos de treinamento. O Capítulo 6 apresenta estudos visando aprimorar e customizar o método de enxame de partículas para aplicação na otimização de *risers*, considerando que os parâmetros deste método tem se mostrado significativamente dependentes da aplicação considerada. No Capítulo 7, são apresentados dois exemplos de aplicação de redes neurais artificiais como metamodelos para análise dinâmica de estruturas. O primeiro exemplo constitui-se de um sistema dinâmico simples (massa-mola-amortecedor) com apenas um grau de liberdade, utilizado para a compreensão dos conceitos envolvidos, e o segundo exemplo constitui-se da resposta dinâmica de linhas de ancoragem conectadas a um FPSO projetado para uma lâmina d'água de 2000m. No Capítulo 8, são apresentados dois exemplos de aplicação de redes neurais artificiais como metamodelos para processos de otimização. O primeiro exemplo baseia-se na aproximação de uma função matemática benchmark (DeJong F1) por uma rede neural treinada a partir de exemplos obtidos na busca pelo mínimo da função. O segundo exemplo baseia-se na aproximação dos parâmetros de resposta da otimização de projeto de um sistema de ancoragem composto por 8 linhas de ancoragem e 5 *risers*, numa profundidade de 2400m. No Capítulo 9 são apresentados os comentários finais acerca da pesquisa desenvolvida e trabalhos futuros.

2 SITEMAS DE EXPLORAÇÃO DE PETRÓLEO OFFSHORE

2.1 Introdução

Para realizar a exploração e a explotação de petróleo no mar é utilizado um conjunto de equipamentos conhecidos como *Sistemas Offshore*. Tais equipamentos podem ser divididos em quatro grupos: casco, linhas, equipamentos submarinos e poços.

Neste capítulo são apresentados em maiores detalhes o casco e, principalmente, as linhas, por serem de fundamental importância no desenvolvimento deste trabalho. No entanto, foge do escopo deste trabalho o detalhamento dos equipamentos submarinos, como árvore de natal molhada, dutos, válvulas, etc. e, também, dos equipamentos de perfuração e completação que constituem os poços de exploração de petróleo.

Tanto a fixação das unidades flutuantes por meio de ancoragem, quanto o transporte de óleo e informações entre o fundo do mar e a unidade flutuante são feitos através de estruturas esbeltas comumente chamadas de linhas. As linhas podem ser ancoragens, *risers*, cabos elétricos e umbilicais. A principal característica das linhas é a sua esbeltez em relação ao seu comprimento. A análise estrutural de tais linhas é complexa devido às grandes não linearidades decorrentes desta esbeltez. Neste trabalho, serão estudadas as linhas de ancoragem e os *risers*. Na seção 2.5 será descrita de forma suscinta as informações principais acerca do projeto de linhas de ancoragem e *risers*.

2.2 Casco

Com o avanço da exploração e produção de petróleo em lâminas d'água mais profundas, além de 200m, tornou-se necessária a utilização de novas unidades de produção. Uma vez que as unidades baseadas em plataformas fixas, utilizadas para profundidades menores, não serviam para estas profundidades, foram desenvolvidas as plataformas flutuantes. Atualmente é possível classificar as plataformas flutuantes em semi-submersíveis, navios, pernas tensionadas (TLP- *Tension Leg Platform*) e *Spar* (Figura 1).

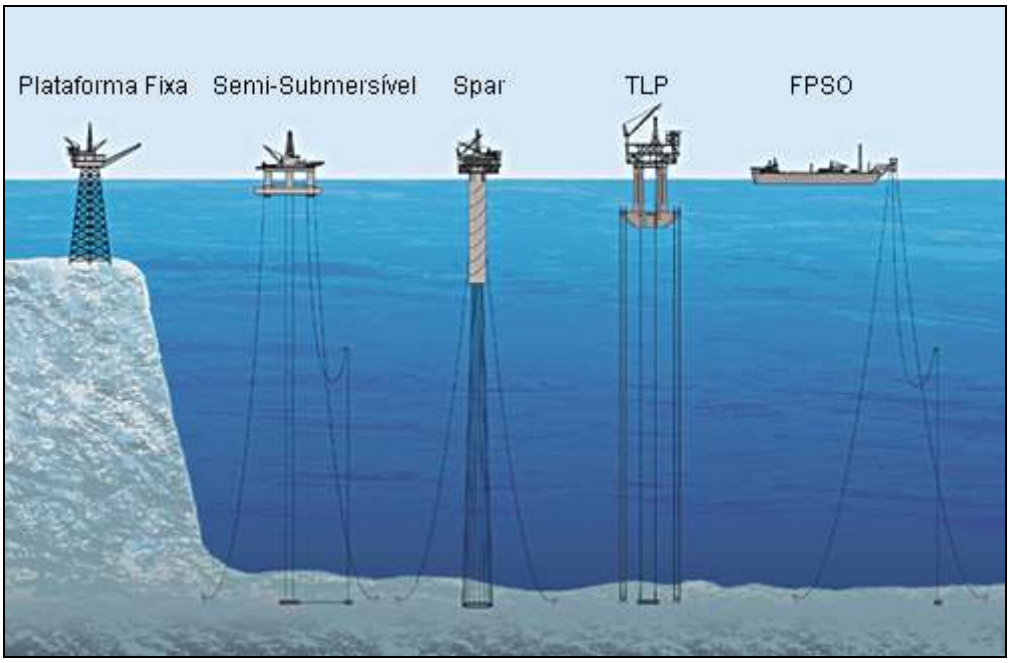

Figura 1. Tipos de plataformas.

2.2.1 Plataforma semi-submersível

As semi-submersíveis são plataformas com estruturas flutuantes largamente empregadas para produção, completação e perfuração (Figuras 2 e 3). São compostas de uma estrutura de um ou mais conveses, apoiada por colunas, também chamadas de pernas, em flutuadores submersos, também denominados de *pontoons*. Uma unidade flutuante sofre movimentações devido à ação das ondas, correntes e ventos, com possibilidade de danificar os equipamentos a serem descidos no poço. Por isso, torna-se necessário que ela fique posicionada na superfície do mar, dentro de um círculo com raio de tolerância ditado pelos equipamentos de subsuperfície, operação esta a ser realizada em lâmina d'água. Dois tipos de sistema são responsáveis pelo posicionamento da unidade flutuante: o sistema de posicionamento dinâmico e o sistema de ancoragem, que será explicitado na seção 2.4. No sistema de posicionamento dinâmico, não existe ligação física da plataforma com o fundo do mar, exceto a dos equipamentos de perfuração. Sensores acústicos determinam a deriva, e propulsores no casco acionados por computador restauram a posição da plataforma.

Devido à pequena área na linha d´água, a plataforma semi-submersível não admite uma variação muito grande de carga sem alterar significativamente o calado. Por isso não é indicada para armazenar o óleo produzido, necessitando de recursos para a exportação deste óleo.

Figura 2. Plataforma semi-submersível.

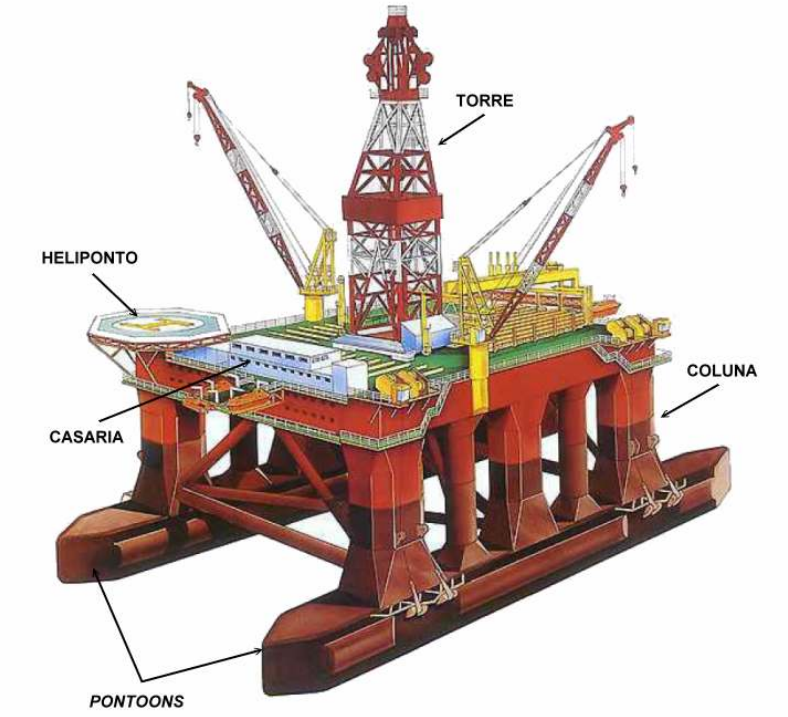

Figura 3. Detalhes da estrutura da plataforma semi-submersível.

2.2.2 Navios

Inicialmente, os navios eram utilizados nas operações de perfuração e/ou completação, sendo denominados navios sonda (Figura 4a). Sua torre de perfuração localiza-se no centro do navio, onde uma abertura no casco permite a passagem da coluna de perfuração.

Posteriormente observou-se que, devido à forma do casco, os navios seriam capazes de armazenar o óleo produzido pelos poços, podendo dispensar um sistema de

exportação contínua. Assim sendo, os navios foram adaptados para extrair, armazenar e exportar petróleo, através da conversão de navios petroleiros ou graneleiros em Unidades de Produção, Armazenamento e Alívio de Petróleo, ou, sua sigla em inglês, FPSO (*Floating Production Storage and Offloading*) (Figuras 4b e 5). Além disso, muitas vezes os navios são utilizados como suporte de outras plataformas para armazenar e exportar óleo. Nestes casos são chamados de FSO (*Floating Storage and Offloading Vessel*).

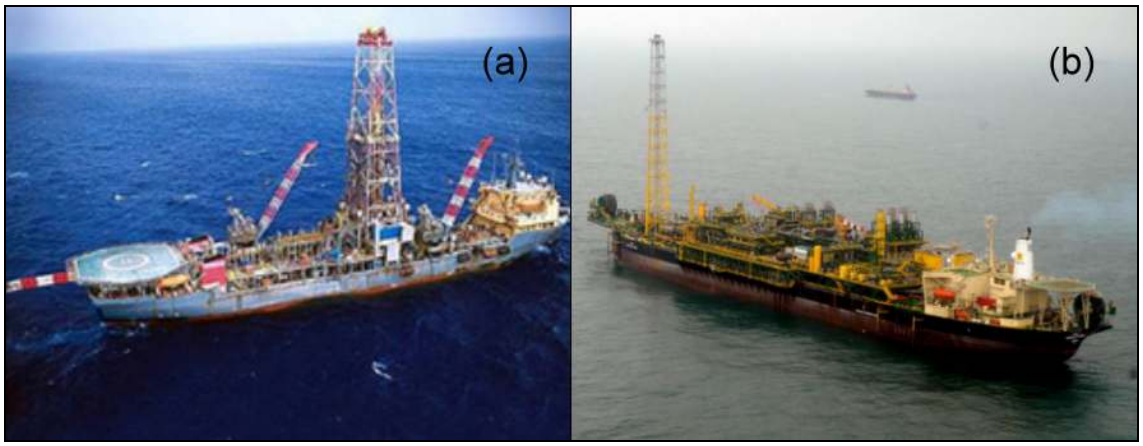

Figura 4. (a) Navio sonda; (b) Unidade FPSO.

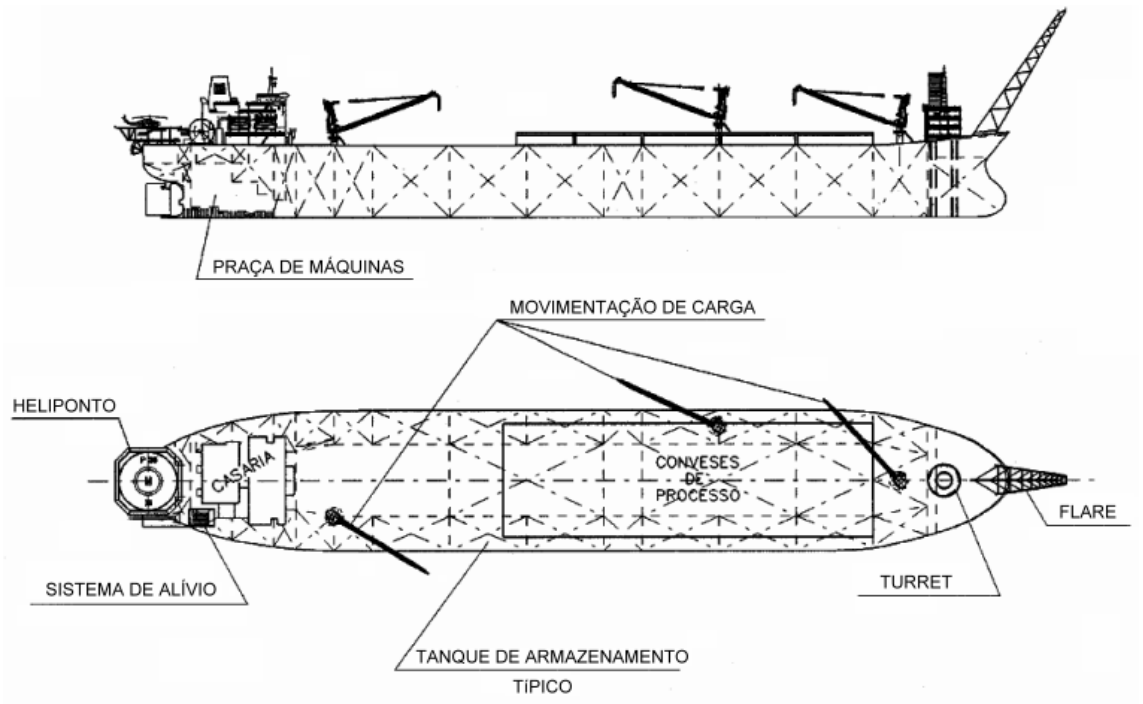

Figura 5. Detalhes da estrutura da unidade FPSO.

2.2.3 Plataforma de Pernas Tensionadas (TLP)

A plataforma TLP (*Tension Leg Platform*) (Figura 6) é uma estrutura flutuante, semelhante à plataforma semi-submersível, cujo excesso de flutuação é compensado através de tendões verticais tensionados (pernas) conectados a um sistema de fundação através de estacas e tracionados no topo pela força resultante entre peso e empuxo (restauração hidrostática).

Comparado a outros tipos de cascos, o casco da TLP possui movimentos mais restritos, o que possibilita que a completação dos poços seja do tipo 'seca', isto é, que o controle e intervenção nos poços seja feito na plataforma e não no fundo do mar, reduzindo assim os custos de instalação e produção do poço.

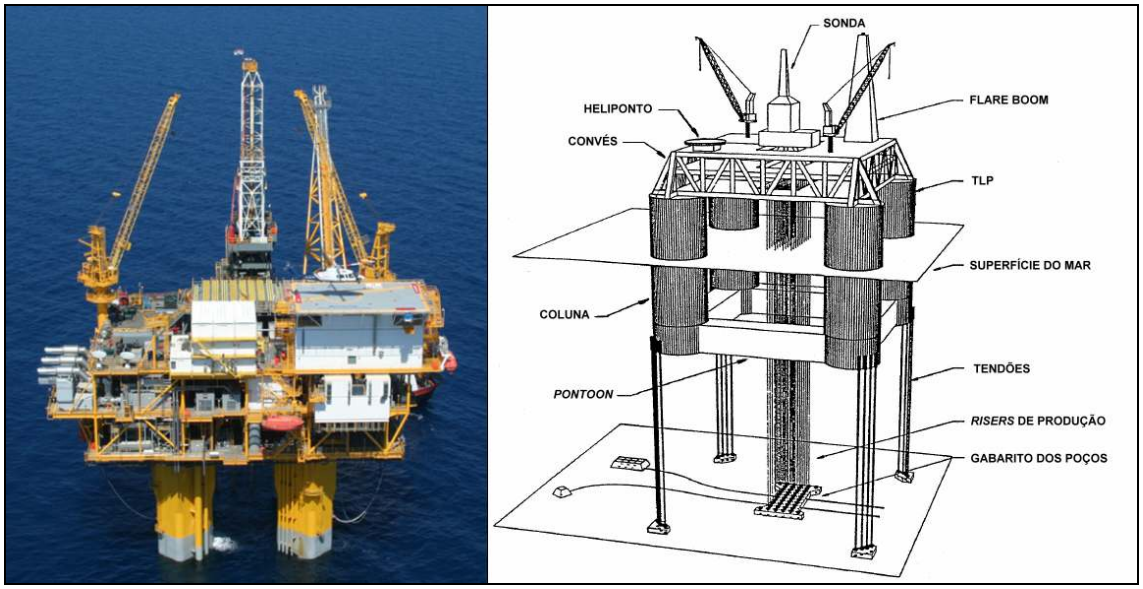

Figura 6. Plataforma TLP e detalhes de sua estrutura.

2.2.4 Plataforma *Spar*

A plataforma do tipo *Spar* é composta de um único cilindro vertical de aço de grande diâmetro, ancorado, com conveses instalados no topo (Figuras 7 e 8). Seu calado tem uma profundidade constante de cerca de 200 m, o que gera pequenos movimentos verticais e possibilita a utilização de *risers* rígidos de produção. Utilizam-se também supressores de vórtices em torno do cilindro com o objetivo de inibir as vibrações induzidas pelo fenômeno de *vortex shedding* ocasionado principalmente pela ação das correntes marinhas.

Figura 7. Plataforma do tipo *Spar*.

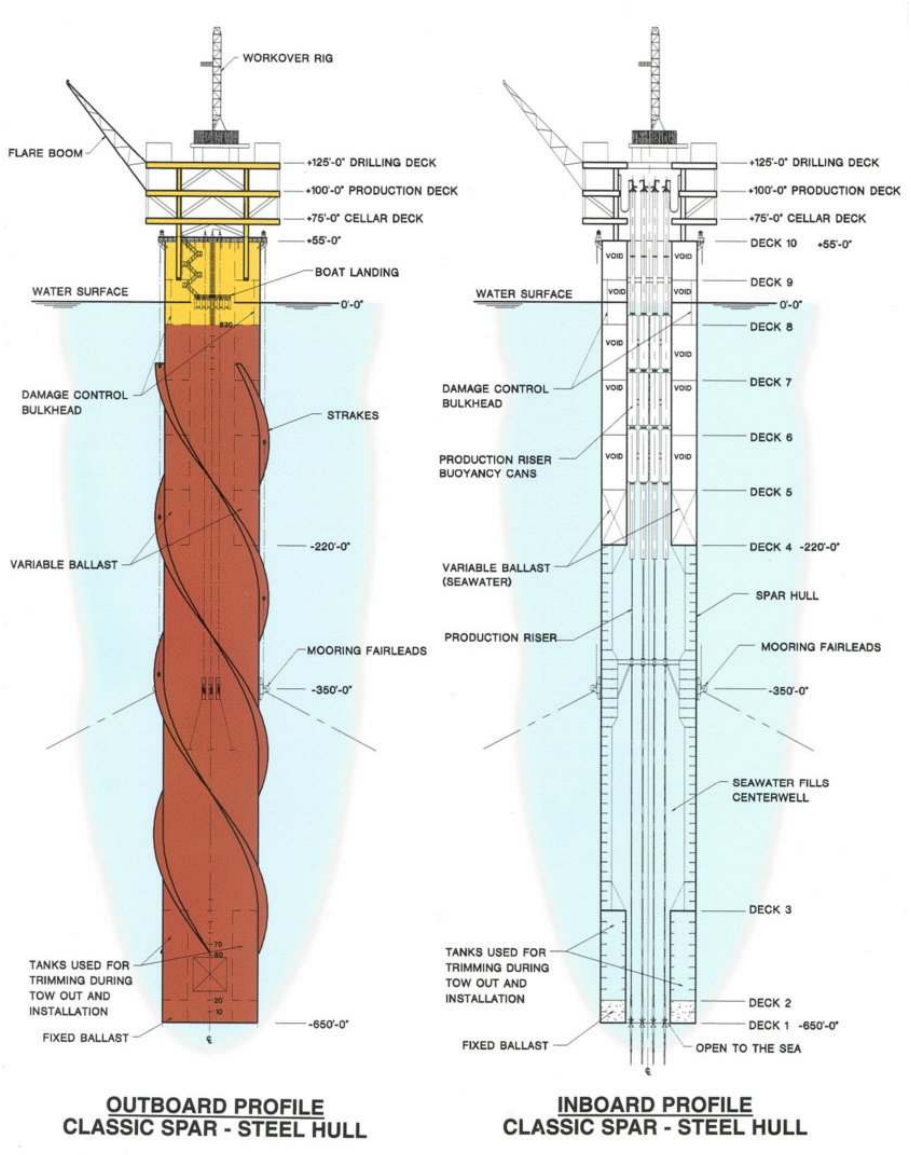

Figura 8. Detalhes da estrutura da plataforma do tipo *Spar*.

2.3 *Risers*

As plataformas flutuantes são posicionadas em locações previamente definidas com base no arranjo de poços do campo de produção, recebendo óleo e/ou gás do poço por meio de dutos denominados *risers*.

Os *risers* são componentes críticos de um sistema submarino de produção, por estarem submetidos a elevados esforços de fadiga devido à ação de correnteza, efeito das ondas e movimentação da unidade flutuante. Portanto, o projeto dos sistemas de *risers* representa um dos principais desafios no desenvolvimento de campos produtivos localizados em grandes lâminas d'água.

Os *risers* podem ser classificados de acordo com a sua configuração, material e finalidade. De acordo com o material, podemos classificar os *risers* em:

Risers Rígidos – Em geral são formados por seqüências de tubos de aço de aproximadamente 12m de comprimento, mas poderiam também ser de titânio ou compósitos. Um exemplo de *riser* rígido pode ser visto na Figura 9. Os *risers* rígidos podem ser aplicados em atividades de perfuração, produção, completação e intervenção. Para serem utilizados em lâminas d'água profundas, podem ser envolvidos por flutuadores para diminuir o seu peso.

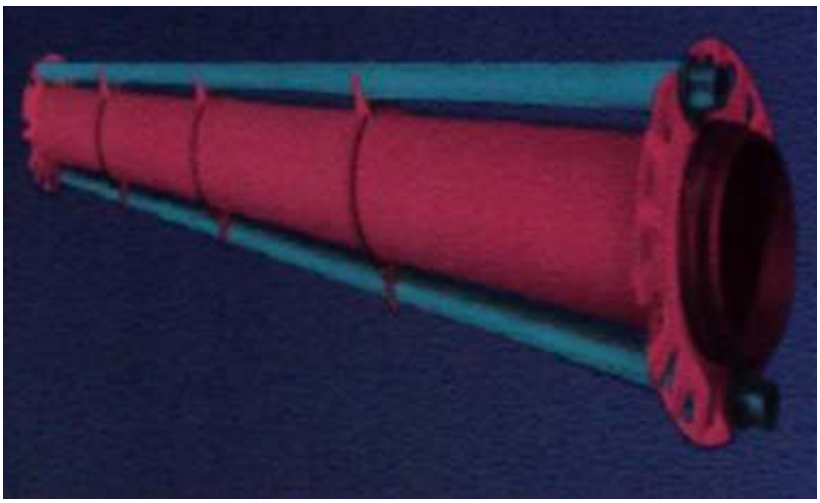

Figura 9. *Riser* rígido.

Risers Flexíveis – Em geral são constituídos de camadas de aço intercaladas com polietileno. As camadas de aço proporcionam flexibilidade ao *riser*, enquanto que as camadas de polietileno proporcionam estanqueidade, proteção contra corrosão e evitam a abrasão das camadas metálicas. A Figura 10 apresenta um esquema básico da composição de camadas de um *riser* flexível.

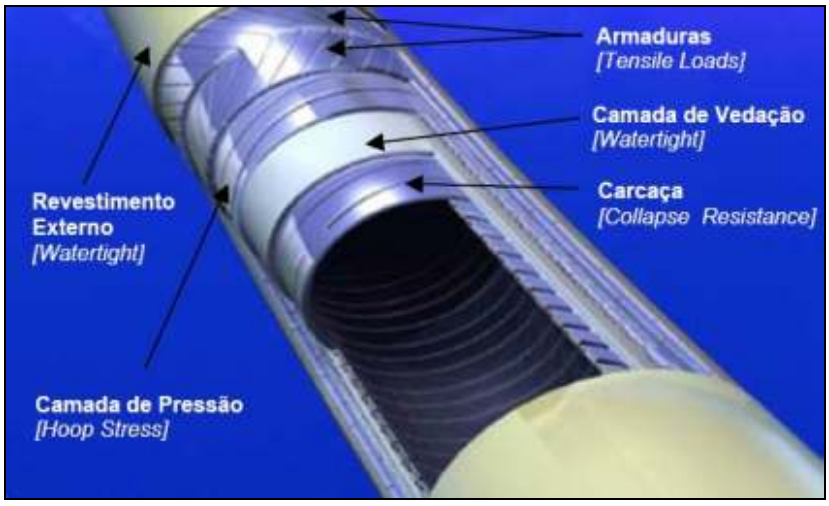

Figura 10. *Riser* flexível.

Os *risers* de perfuração e completação apresentam geometria vertical. Já os *risers* de produção ou injeção podem assumir diferentes configurações em catenária, como segue:

Catenária Livre – Os *risers* configurados em catenária livre (*free-hanging catenary*) são de fácil instalação, porém apresentam maiores esforços na conexão com a plataforma. Esse tipo de configuração pode ser utilizado tanto em *risers* rígidos como em flexíveis. A Figura 11 exemplifica uma configuração em catenária livre.

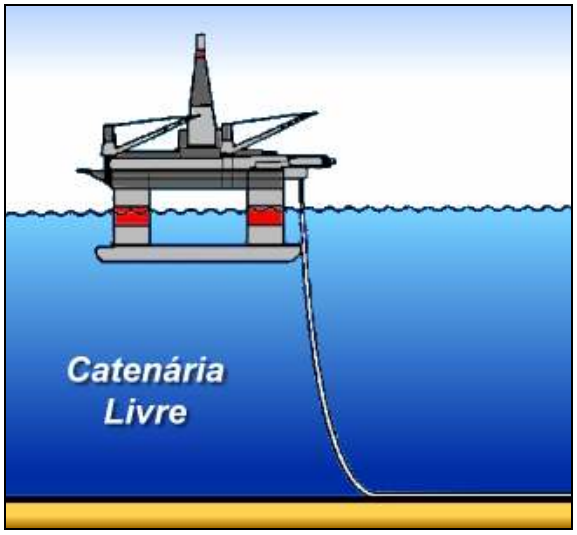

Figura 11. Configuração em Catenária Livre.

Lazy-S **e** ***Steep-S*** – Esses tipos de configurações de *risers* apresentam uma seção intermediária que passa por um arco com flutuadores, cujo empuxo alivia o peso suportado pelo sistema flutuante, e contribui com o momento restaurador quando sob

solicitações laterais. Na configuração *Lazy-S* (Figura 12a), há um tensionador sustentando o arco flutuador. Já na configuração *Steep-S* (Figura 12b), o próprio *riser* tensiona o arco flutuador.

Lazy Wave* e *Steep Wave – Esses tipos de configurações de *risers* têm comportamento semelhante às configurações *Lazy-S* e *Steep-S*, mas o arco é substituído por uma seção intermediária com flutuadores distribuídos, simplificando a instalação. Os desenhos nas Figuras 12c e 12d exemplificam estas configurações, respectivamente.

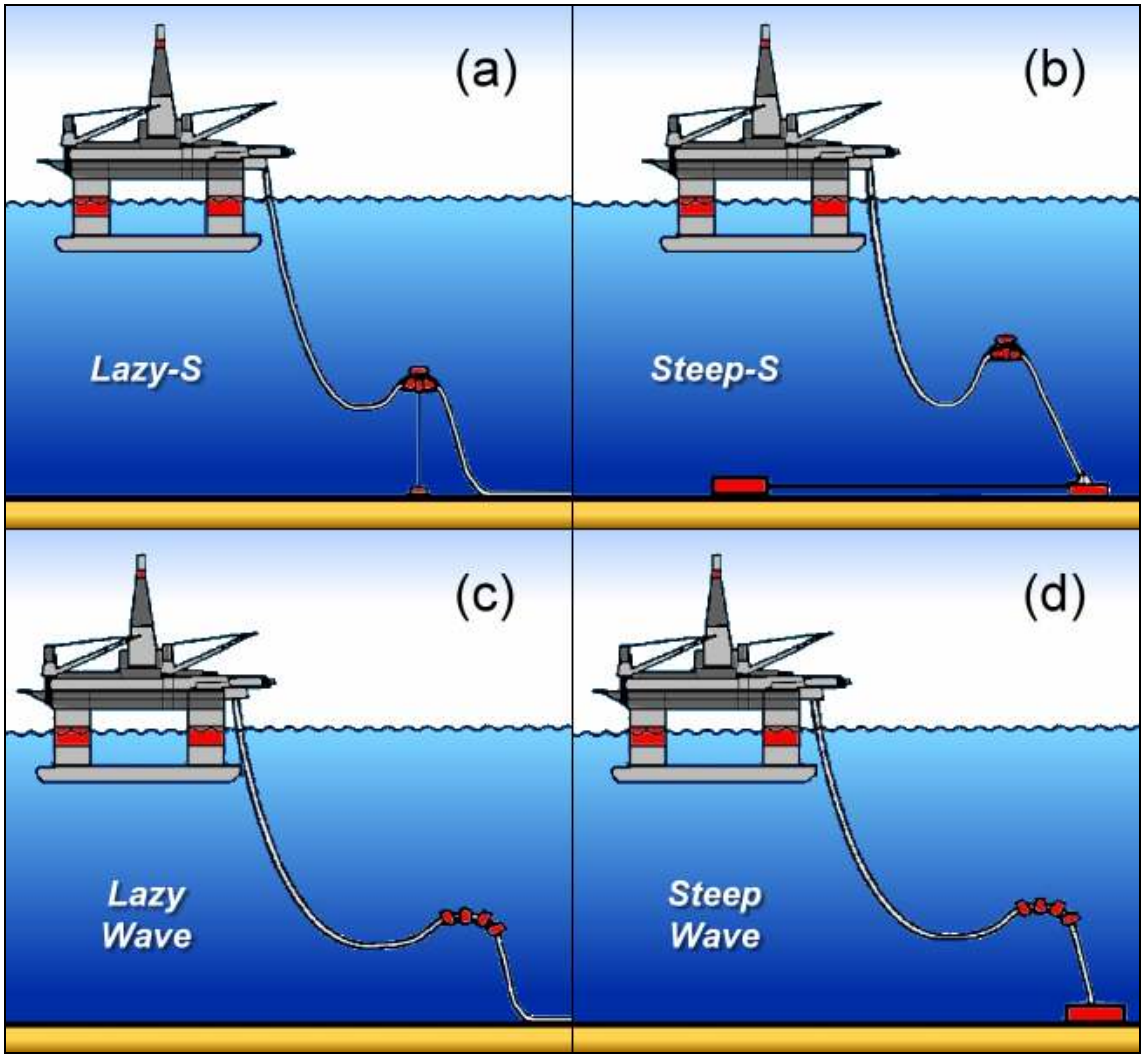

Figura 12. Configurações de *risers* com flutuadores.

Outras configurações de *risers* são possíveis e, além disso, podem ser utilizados sistemas híbridos ou mistos, criados a partir de uma combinação de *risers* rígidos e flexíveis, baseados em bóias submersas (Figura 13).

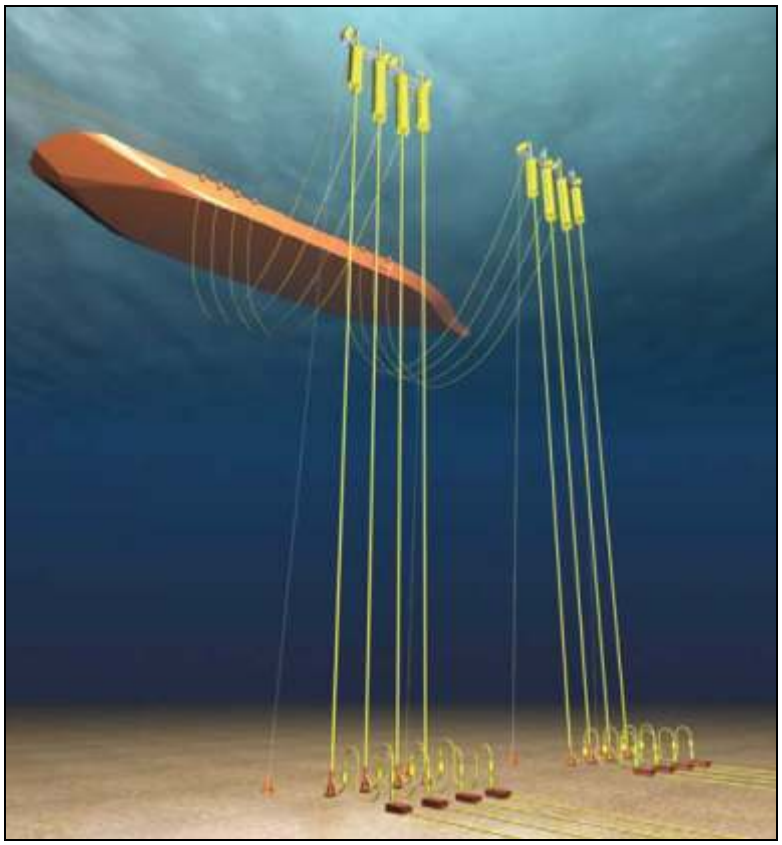

Figura 13. Sistema híbrido de *risers*.

Com as descobertas de poços petrolíferos a profundidades acima de 1000 m, o uso do conceito de *riser* flexível apresentou dificuldades em decorrência da grande pressão hidrostática e das cargas dinâmicas. Os *risers* rígidos sobrepujam este problema com simples aumento de espessura de parede e apresentam a vantagem de empregar materiais mais baratos e de proporcionarem um processo de fabricação mais simples. O *riser* rígido em catenária (*Steel Catenary Riser* – SCR) representa uma alternativa ao uso dos *risers* flexíveis de grande diâmetro em decorrência de aspectos técnicos e econômicos, o que também influencia os custos dos sistemas de coleta e exportação.

A utilização de *risers* rígidos em catenária é considerada uma alternativa disponível para qualquer aplicação em plataformas semi-submersíveis. O interesse na aplicação de *risers* rígidos em catenária conectadas a FPSOs (unidades flutuantes de produção, armazenagem e descarga), em decorrência da tendência de usar essas unidades para exploração e produção em águas profundas, trouxe a necessidade de se estudar esse conceito cuidadosamente, devido aos grandes deslocamentos impostos pelo navio no topo do *riser*, principalmente o movimento de heave.

2.4 Linhas de Ancoragem

As linhas de ancoragem têm a função estrutural de fornecer forças de restauração para manter em posição os sistemas flutuantes, tais como, plataformas semi-submersíveis ou navios.

São estruturas esbeltas dispostas em catenária (ancoragem convencional) ou utilizadas como linhas retesadas (*taut-leg*) ou tendões.

Os navios e as plataformas semi-submersíveis podem ser ancorados pelas suas extremidades. No caso do navio, a ancoragem pode ser central, o que permite que o corpo flutuante se oriente em relação à força ambiental predominante (AMERICAN BUREAU OF SHIPPING, 1996).

Os materiais mais utilizados nas linhas de ancoragem são as amarras de aço, os cabos de aço e mais recentemente os cabos de poliéster (ROSSI *et al.*, 2001).

Geralmente utilizam-se amarras nos trechos iniciais e finais das linhas de ancoragem uma vez que este material é mais resistente ao manuseio e ao atrito com o fundo e com os guinchos das plataformas.

2.4.1 Ancoragem Convencional

A ancoragem convencional é aquela cujas linhas possuem a forma de catenária (Figura 14). Esta técnica de ancoragem é utilizada em operações de produção ou perfuração com a vantagem de possibilitar maiores passeios da embarcação sem a necessidade de âncoras com elevado poder de garra, mantendo a unidade flutuante estacionária através da força de restauração das linhas. As linhas ancoradas são presas ao fundo do mar por âncoras de resistência horizontal.

Neste tipo de ancoragem, o próprio trecho de linha apoiado no fundo contribui com a força de restauração através do atrito com o fundo, aliviando os esforços nas âncoras em condições normais de operação. No entanto, o custo da linha aumenta devido à necessidade deste grande trecho no fundo, geralmente de amarra.

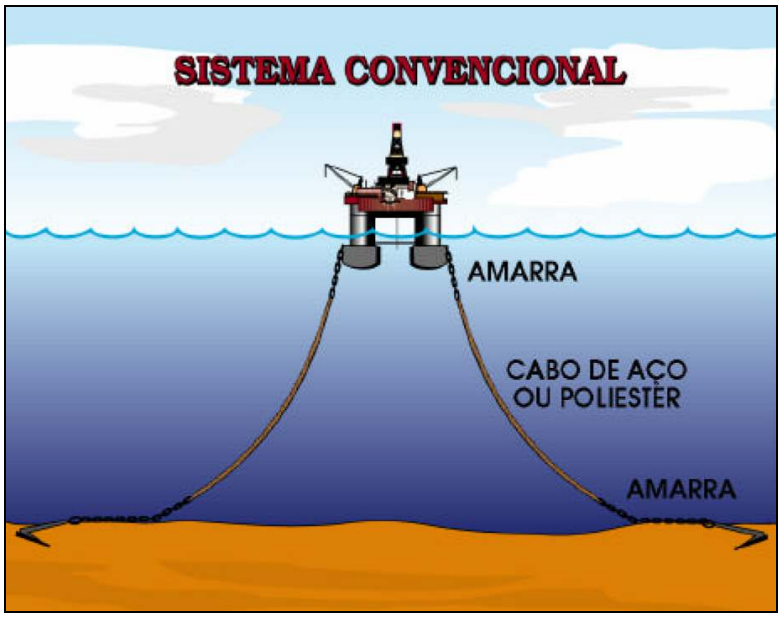

Figura 14. Sistema de ancoragem convencional (GONÇALVES & COSTA, 2002).

Um dos principais problemas deste tipo de ancoragem é que a força de restauração está relacionada ao raio de ancoragem que, para atender aos critérios de projeto para passeio das unidades flutuantes ancoradas, deve ser razoavelmente grande (10% da lâmina d'água). Conseqüentemente, em um campo de exploração de petróleo, isto gera um congestionamento entre as linhas de ancoragem e linhas de unidades próximas, interferindo diretamente no posicionamento das mesmas, além de poder causar a interferência das linhas com dutos e equipamentos submarinos.

Uma plataforma semi-submersível é menos sensível à direcionalidade dos carregamentos ambientais do que um navio, por isso é mais utilizada a distribuição simétrica das linhas de ancoragem em torno da embarcação. Esta configuração é conhecida como SMS (*Spread Mooring System*).

Em contrapartida, devido à forma do casco, os navios tendem mais a se alinhar com os carregamentos de correnteza, onda e vento. Assim, com o objetivo de evitar uma sobrecarga no sistema de ancoragem, utiliza-se um sistema no qual as linhas de ancoragem e os *risers* são conectados a uma estrutura em forma de torre. O casco da embarcação é conectado a esta estrutura através de um anel deslizante que faz com que ele gire em torno de um ponto limite, a torre (*turret*), para poder se alinhar com os carregamentos ambientais. Este sistema é conhecido como SPM (*Single Point Mooring*) (Figura 15).

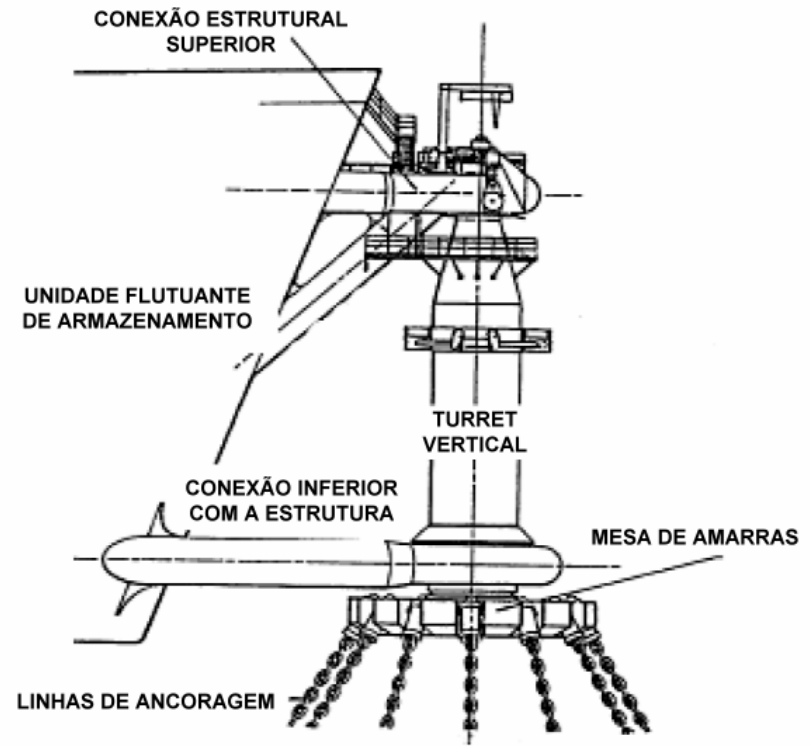

Figura 15. *Single Point Mooring* com turret externo.

2.4.2 Linhas Retesadas (*Taut-leg*)

Para contornar as desvantagens do sistema em catenária, pode-se adotar a ancoragem *taut-leg* (Figura 16), que consiste em linhas mais retesadas com um ângulo de topo de aproximadamente 45° com a vertical, tendo assim uma projeção horizontal menor, para uma profundidade de lâmina d'água da mesma ordem de grandeza.

Este tipo de ancoragem proporciona maior rigidez ao sistema, sendo o passeio da embarcação limitado a *offsets* menores (distâncias menores). No entanto, o uso do sistema *taut-leg* exige que as âncoras utilizadas suportem esforço vertical, por isso são utilizadas estacas de sucção, VLA (*Vertically Loaded Anchors*) ou estacas de fundeio para a fixação das linhas ao fundo.

As linhas da ancoragem *taut-leg* são constituídas por cabos de aço ou amarras nas suas extremidades e cabo de poliéster ou aço no seu trecho intermediário.

A ancoragem *taut-leg* é geralmente utilizada em sistemas localizados em regiões de grandes profundidades, pois a ancoragem em catenária demandaria um raio de ancoragem muito grande.

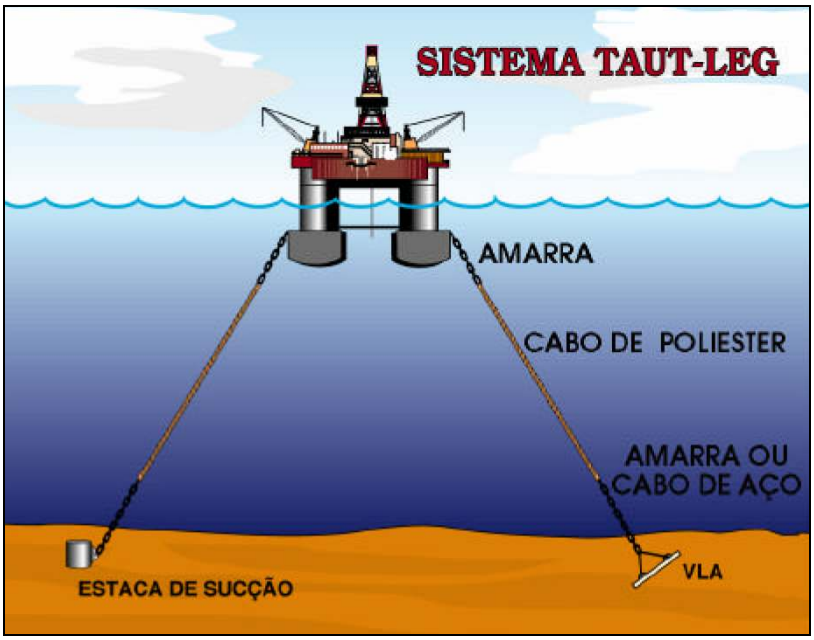

Figura 16. Sistema de ancoragem *Taut-Leg* (GONÇALVES & COSTA, 2002).

2.4.3 Tendões

Os tendões são linhas de ancoragem verticais que permanecem constantemente tracionadas devido ao excesso de empuxo proveniente da parte submersa da embarcação. Podem ser constituídos de cabo de aço ou de material sintético, proporcionando uma elevada rigidez no plano vertical e baixa rigidez no plano horizontal.

A força de restauração no plano horizontal é fornecida pela componente horizontal da força de tração nos tendões. Para tendões de pequenos diâmetros ($d \cong 0{,}25m$), os efeitos de flexão podem ser desprezados, enquanto que, para grandes diâmetros ($d \cong 1{,}00m$), tais efeitos devem ser considerados (JUVINAO CARBONO, 2005).

Utiliza-se este tipo de ancoragem em sistemas que proporcionem um excesso de flutuabilidade capaz de manter os tendões permanentemente tracionados, como as plataformas do tipo TLP, além de também poder ser adotada por bóias de sub-superfície.

2.4.4 Complacência Diferenciada (DICAS)

A partir da experiência adquirida na operação de um FSO no campo de Caravelas na Bacia de Campos em 1993, foi desenvolvido pela Petrobras um novo tipo de ancoragem, denominado DICAS (*Differentiated Compliance Anchoring System*), que consiste num sistema com complacência diferenciada entre as linhas de popa e proa da embarcação (GARZA-RIOS *et al.*, 1997; MASSETTI, 1997).

As linhas de popa oferecem uma resistência menor aos movimentos da embarcação, permitindo que a mesma se alinhe com as cargas ambientais dentro de um limite angular (5 a 7 graus). Apesar do custo menor, o sistema só pode ser empregado em locações com pequena variação angular média da resultante do carregamento (ALBRECHT, 2005).

2.4.5 Âncoras

A âncora é o elemento responsável pela fixação da linha de ancoragem a um determinado ponto do fundo do mar. As linhas de ancoragem, a fim de manter a posição horizontal da unidade flutuante, transmitem os esforços do sistema para este ponto, denominado ponto de ancoragem.

O ponto de fixação dependerá da forma de ancoragem utilizada, no que diz respeito à capacidade de resistir a cargas verticais.

Na ancoragem convencional, com as linhas em catenária, a âncora deve resistir apenas a cargas horizontais, sendo mais comum a utilização de âncoras de arraste (Figura 17), que são de fácil instalação. No entanto, as âncoras deste tipo necessitam de um longo trecho de linha apoiado no fundo para garantir que não sofrerão cargas verticais.

Em contrapartida, na ancoragem com linhas retesadas, a âncora deverá suportar cargas verticais de grande magnitude. Para este fim podem ser usadas as âncoras VLA (*Vertically Loaded Anchors*) ou estacas de fundeio.

As estacas mais utilizadas são:

- Estacas de sucção, que são fechadas no topo e possuem grande diâmetro e pequeno comprimento (COLLIAT, 2002), e são fixadas no fundo do mar retirando-se água do seu interior, o que cria uma diferença de pressão que faz com que ela se enterre (Figura 18).

- Estacas tradicionais (cravadas por percussão), que possuem grande comprimento e pequeno diâmetro, e são instaladas com martelo hidráulico submarino.
- Estacas torpedo, que são estacas propriamente ditas, e são lançadas de uma determinada altura (~100m) para que sejam cravadas no fundo por intermédio da força de gravidade, de forma que a própria energia da queda faz com que a estaca atinja a penetração necessária no solo.

Um estudo mais detalhado sobre âncoras pode ser encontrado em (ELTAHER *et al.*, 2003).

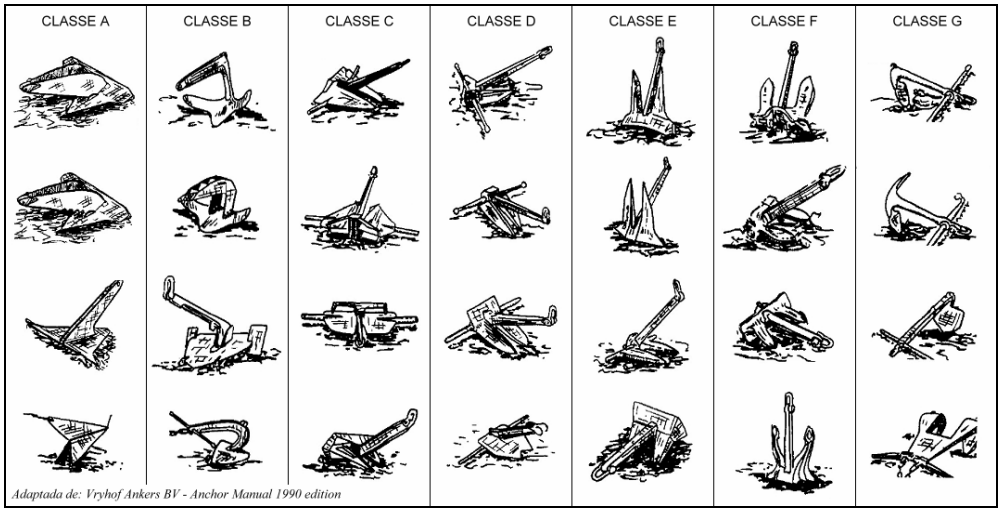

Figura 17. Tipos de âncora de arraste.

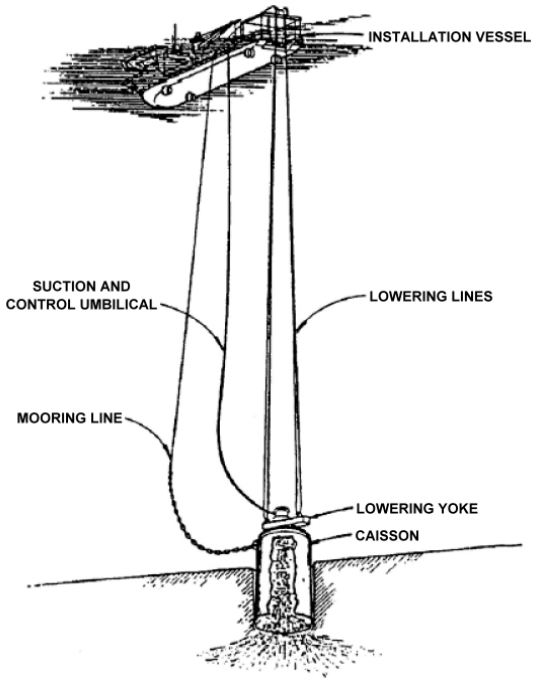

Figura 18. Estaca de Sucção (API, 2001).

2.4.6 Amarras

As amarras são correntes flexíveis formadas pela conexão de múltiplos elos feitos de metal rígido.

O tipo de amarra mais comumente utilizado na ancoragem de plataformas são as que possuem elos com malhete (ALBRECHT, 2005). O malhete nos elos assegura que estes ficarão na posição correta nos guinchos de tambor.

De acordo com a sua forma, dimensões e diâmetro, o elo pode ser classificado como:

- Elo Comum (*Common Link*) – Possui um reforço interno (malhete) que ajuda a enrijecer a corrente feita com tais elos e evita que ela fique emaranhada. Utilizado na amarra para sua formação propriamente dita.
- Elo Alargado (*Enlarged Link*) – Semelhante ao elo comum quanto à forma, só que com dimensões maiores. Empregado no acabamento da amarra e especial para receber partes de diâmetro, no máximo, igual ao seu.
- Elo Final (*End Link*) – Elo tipo olhal, sem malhete, usado na formação do chicote da amarra.

As dimensões de um elo são múltiplos do diâmetro da corrente (Figura 19).

Existem diversos tipos de componentes utilizados para unir duas partes de corrente, dos quais o mais comumente empregado é o elo Kenter (Figura 20). Apesar destes elementos possuírem uma carga de ruptura igual e, em certas ocasiões, superior a de uma corrente de mesma dimensão, a durabilidade à fadiga é consideravelmente menor. Por este motivo, as linhas de ancoragem devem utilizar o menor número possível destes elos de ligação.

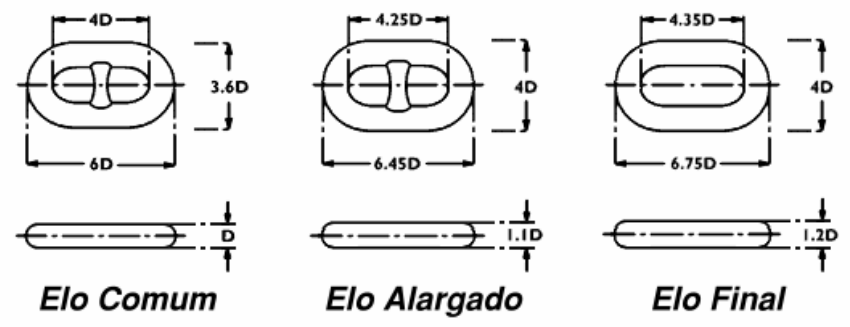

Figura 19. Tipos de elo de amarra.

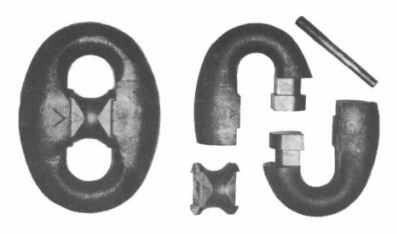

Figura 20. Elo Kenter e sua vista desmontada.

2.4.7 Cabos de Aço

Um cabo de aço consiste de 3 elementos: os arames, as pernas e a alma (Figura 21). Os arames metálicos constituem o elemento básico de um cabo de aço. Enrolando helicoidalmente um determinado número de arames em torno de um arame central, forma-se uma perna (*strand*). São as pernas, que enroladas helicoidalmente em torno de um núcleo, a alma (*core*), vão formar o cabo de aço. As pernas suportam a maior parte da carga. A alma tem como função principal suportar as pernas em condições de flexão combinadas a tração.

A especificação do cabo é feita pelo número de pernas da última camada e o número de arames formadores das pernas da última camada. Por exemplo: 6 x 19 *Filler-Wire*, tem 6 pernas e a última camada de perna é formada por 19 arames.

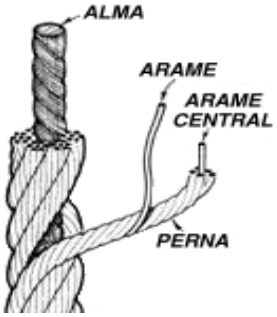

Figura 21. A estrutura do cabo de aço.

Os tipos de cabos de aço mais utilizados na ancoragem de unidades flutuantes são o *six strand* e o *spiral strand*. Os cabos *six strand* são empregados com maior frequência em unidades de perfuração, devido ao seu fácil manuseio. Já os cabos *spiral strand* com torque balanceado são melhores que os cabos *six strand* e podem ter uma durabilidade muito grande quando são encapados com uma camada plástica. A desvantagem deste tipo de cabo é que ele é mais caro e deve ter um raio de curvatura mínimo (MBR) de aproximadamente 22×D, onde D é o diâmetro externo do cabo. Esta

limitação dificulta o manuseio e a utilização em guinchos de tambor, o que torna a sua aplicação mais adequada para unidades de produção do que para unidades de perfuração (ALBRECHT, 2005). A Figura 22 ilustra os tipos de cabos de aço mais utilizados nos sistemas de ancoragem.

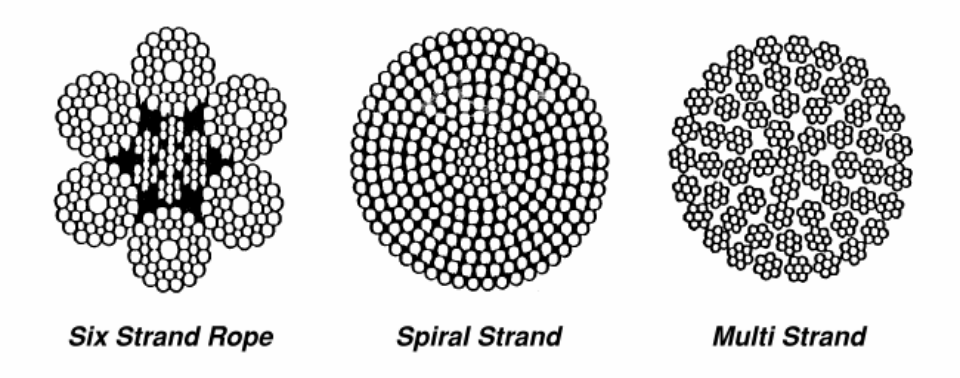

Figura 22. Cabos de aço (API, 1997).

Os cabos de aço possuem um peso relativamente menor do que as amarras, mas uma elasticidade maior para a mesma carga de ruptura. No entanto, a desvantagem dos cabos de aço é que estes são mais vulneráveis à corrosão do que as amarras. Para minimizar a corrosão da trança metálica dos cabos de aço podem ser empregados cabos galvanizados. Mesmo assim, atualmente, a vida útil de um cabo de ancoragem é bem inferior à vida útil de uma amarra.

Com relação à resistência dos arames que formam o cabo, são utilizados normalmente os tipos IPS (*Improved Plow Steel*) e EIPS (*Extra Improved Plow Steel*). Os cabos com este último tipo de arame são mais resistentes à tração e, portanto, mais recomendados nos sistemas de unidades flutuantes.

2.4.8 Cabos de Poliéster

Cabos de fibra e cordas são formados por fios naturais ou sintéticos, torcidos e retorcidos helicoidalmente, utilizados para tração. Em geral, classificam-se como cabos todos os cabos sintéticos, e como cordas, todos os cabos e cordas feitos de qualquer fibra natural.

As fibras mais utilizadas para a fabricação dos cabos são: polietileno, sisal, poliamida (comercialmente conhecida como *nylon*), polipropileno, poliaramida (cuja fibra possui grande módulo de elasticidade e tem nomes comerciais tais como *Kevlar*, *twaron* e *technora*), *HMPE* (*High Modulus Polyethylene*, comercialmente conhecido como *dyneema* ou *spectra*) e poliéster.

O poliéster é o cabo sintético mais utilizado em sistemas de ancoragem de plataformas, e espera-se que possa atingir uma vida útil de até 20 anos. A Figura 23 mostra um diagrama com os detalhes de um típico cabo de poliéster utilizado em ancoragens de unidades flutuantes.

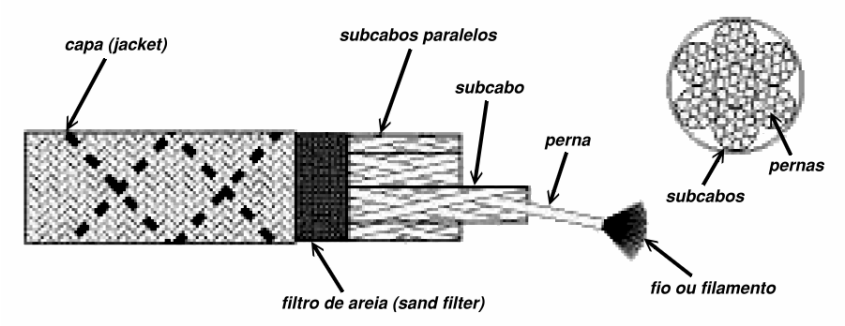

Figura 23. Diagrama do cabo de poliéster (GONÇALVES & COSTA, 2002).

Para que a construção do cabo de poliéster seja de alta eficiência, é necessário que as fibras sejam mantidas, tanto quanto possível, posicionadas formando pequenos ângulos em relação ao eixo do cabo. Atualmente, para atender a este requisito, três construções de cabos estão sendo fabricadas: fios paralelos, subcabos paralelos e tipo cabo de aço.

A resistência específica (*carga de ruptura/massa por unidade de comprimento*) e a rigidez específica ((*carga/massa por unidade de comprimento)/deformação*) de cabos de poliéster com as três construções mencionadas acima são muito semelhantes. Para as construções de subcabos paralelos e tipo cabo de aço, isto se deve ao uso de configurações de baixa torção. Para o cabo de fios paralelos, isso é explicado pela dificuldade em aproveitar-se todo o potencial do fio, devido à variação de propriedades entre filamentos dentro do fio e às diferenças no comprimento dos fios (ALBRECHT, 2005).

O cabo de poliéster apresenta uma flexibilidade axial bem maior que a do cabo de aço e das amarras com a mesma carga de ruptura nominal e um peso submerso por unidade de comprimento bem menor. Eles providenciam ainda a complacência necessária em decorrência das propriedades elásticas da fibra, evitando o uso de longas configurações em catenária usadas em sistemas de amarração convencionais (MONTEIRO, 2008).

Os tendões de poliéster devem estar necessariamente submetidos a esforços axiais de tração, pois apresentam baixa ou nenhuma rigidez à compressão axial.

2.5 Projeto de Linhas de Ancoragem e *Risers*

A definição completa de critérios práticos que levem a um projeto seguro de linhas de ancoragem e *risers* é um assunto bastante complexo e que remete a distintas áreas da engenharia estrutural.

O projeto da ancoragem deve ser desenvolvido de forma a garantir que o passeio da unidade flutuante, mesmo sob a condição de tempo mais severa, não danifique a estrutura dos *risers* conectados à plataforma, no caso de uma unidade de produção, ou não prejudique a operação de uma unidade de perfuração. O passeio é a distância horizontal que a unidade percorre desde a sua posição de equilíbrio neutro até a posição de equilíbrio sob o carregamento ambiental. Este passeio é normalmente medido como um percentual da lâmina d'água (ALBRECHT, 2005).

Quanto mais rígido for o sistema de ancoragem menor será o passeio. No entanto, o tipo de material utilizado nas linhas de ancoragem determinará o limite superior para a rigidez do sistema, pois aumentar a rigidez implica em aumentar as trações aplicadas às linhas, aumentando consequentemente a tensões internas. Estas tensões devem permanecer dentro de um intervalo de segurança (API, 1997).

Além disso, a composição do sistema de ancoragem deve ser tal que garanta a integridade do mesmo durante o tempo de operação da unidade. Ou seja, o sistema não deve falhar por fadiga.

Os *risers* devem ser analisados para assegurar níveis aceitáveis de deformações, tensões e resistência à fadiga, devido às forças impostas pelas correntes, ondas e movimentos da embarcação. A pressão hidrostática interna (fluido de perfuração) e a externa (água do mar) também são fundamentais nas análises, assim como a influência do fluxo de corrente e onda ao redor do tubo.

A determinação do ponto ótimo de operação das linhas de ancoragem e dos *risers* é feita através do projeto de tais estruturas, considerando-se os carregamentos atuantes nas mesmas.

2.5.1 Considerações de Carregamento

O conjunto de carregamentos possíveis atuantes em linhas de ancoragem e *risers* é dividido em três subgrupos: carregamentos funcionais, carregamentos ambientais e carregamentos acidentais.

- ***Carregamentos funcionais*** – são aqueles que são conseqüência da existência do sistema e não levam em consideração os efeitos ambientais ou acidentais. Como exemplos de carregamentos funcionais temos: peso das linhas, pressão interna do *riser*, pressão hidrostática externa, tração no topo das linhas e inércia.
- ***Carregamentos ambientais*** – são aqueles impostos direta ou indiretamente pelo ambiente oceânico. Como exemplos de carregamentos ambientais temos: vento, força da onda, força das correntes e movimentos da embarcação (Figura 24).
- ***Carregamentos acidentais*** – são aqueles que resultam de ocorrências não planejadas, relacionadas a condições anormais de operação, falhas técnicas ou falhas humanas. Como exemplos de carregamentos acidentais temos: terremotos, explosões, rompimento de uma linha de ancoragem, colisão entre *risers*, entre outros.

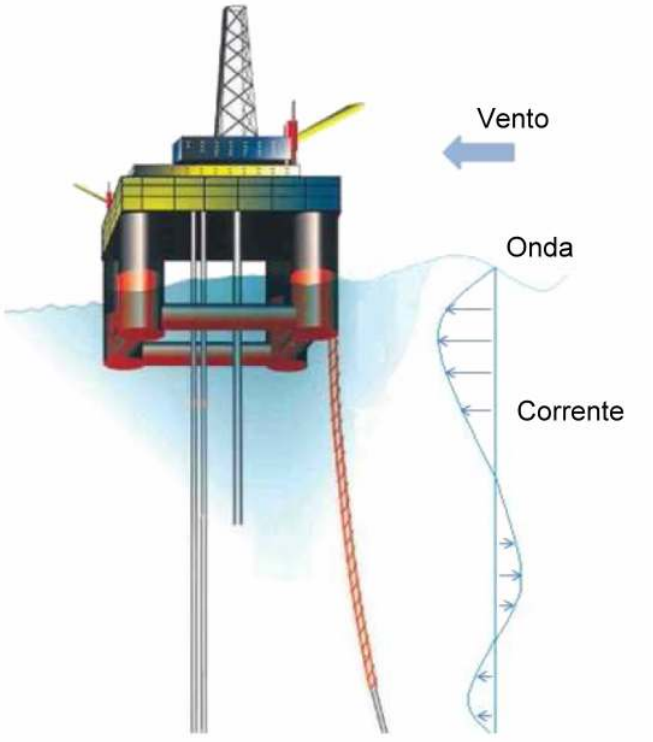

Figura 24. Ações ambientais impostas nos sistemas (MARTINS, 2008).

O projeto completo de sistemas de ancoragem e de *risers* deve considerar um conjunto mínimo de casos de carregamento de modo que seja possível a correta avaliação da resposta estrutural dos sistemas para fazer a comparação com os limites admissíveis para as condições de tensão máxima, fadiga, carregamento máximo para componentes específicos, etc.

Os casos de carregamento devem dar minimamente a noção do comportamento das linhas de ancoragem e dos *risers* ao longo de sua vida em utilização, portanto, considera-se uma combinação de carregamentos funcionais, ambientais e acidentais.

2.5.2 Métodos de Análise

Um problema de análise global de linhas de ancoragem e *risers* passa pela definição das características físicas e geométricas do sistema estrutural, bem como as cargas a que o mesmo está sujeito, além dos movimentos impostos pelo corpo flutuante às linhas. O objetivo das análises é o de monitorar os esforços e as deformações impostas às linhas pelos carregamentos atuantes no sistema.

A ferramenta mais adequada para a análise do comportamento estrutural de linhas de ancoragem e *risers* é a simulação numérica, através da qual se analisa um modelo matemático que representa o problema. Este modelo matemático, que determina um *Problema de Valor Inicial e de Contorno* (PVI/C), é composto por um sistema de equações diferenciais parciais e um conjunto de condições de contorno no espaço e condições iniciais no tempo.

Em casos gerais, não é possível obter uma solução *analítica* para este PVI/C com um grau de precisão aceitável. Por esse motivo, deve-se empregar um método numérico de análise, substituindo-se o modelo matemático *contínuo* (definido em um número infinito de pontos internos da estrutura) por um modelo numérico *discreto* (definido em um número finito de pontos internos). Este método numérico de análise é chamado de *Método dos Elementos Finitos* (MEF).

Devido às características tanto do ambiente, quanto do próprio sistema, as estruturas de linhas de ancoragem e de *risers* apresentam um comportamento não-linear acentuado. Nas análises de tais linhas devem ser consideradas não-linearidades geométricas, provenientes da magnitude dos deslocamentos; não-linearidades físicas devido às relações constitutivas não-lineares dos materiais; não-linearidades das condições de contorno (contato e cargas não-conservativas); entre outras.

Ainda levando em consideração as características do sistema e do carregamento ao qual ele se submete, as análises de estruturas de linhas de ancoragem e de *risers* devem contemplar efeitos estáticos e/ou dinâmicos, dependendo das ações atuantes no sistema:

- ***Ações permanentes*** – peso próprio, empuxo, correnteza e parcela estática do movimento da unidade flutuante imposto no topo das linhas ("*offset*" estático, devido à correnteza e às cargas de deriva média das ondas). Tais ações possuem comportamento estático e, portanto, devem ser analisadas por meio de uma análise estática.
- ***Ações variáveis*** – onda e parcela dinâmica do movimento da unidade flutuante (freqüência de onda e deriva lenta). O tipo de análise empregado em ações variáveis depende da velocidade que ocorrem essas variações:
  - Ações que variam "lentamente com o tempo" – apresentam comportamento "quasi-estático" e, portanto, devem ser analisadas por meio de uma análise estática.
  - Ações variando com um período próximo ou menor que o maior período natural da estrutura – apresentam comportamento dinâmico e, portanto, devem ser analisadas por meio de uma análise dinâmica.

Tais análises podem ser feitas por técnicas de discretização no domínio do tempo ou no domínio da freqüência. As análises no domínio do tempo são bastante dispendiosas computacionalmente, mas podem tratar com rigor as não-linearidades típicas do comportamento de sistemas de ancoragem e de *risers*. Já, as análises no domínio da freqüência, embora menos dispendiosas, ignoram os efeitos não-lineares ou os tratam de forma aproximada através de *técnicas de linearização*.

Além disso, deve ser dada uma atenção especial aos problemas de convergência que são inerentes ao tratamento numérico do problema. Desta forma, é fácil perceber que, para alcançar resultados consistentes a partir de métodos numéricos, torna-se necessária a utilização de técnicas que certamente irão resultar em um custo computacional elevado.

Os programas de simulação numérica tradicionalmente utilizados na prática dos projetos de unidades flutuantes ancoradas baseiam-se em procedimentos "desacoplados". Recentemente, pesquisadores têm reconhecido que os projetos de sistemas flutuantes deveriam considerar uma metodologia "acoplada" que leva em conta a completa interação da embarcação com as linhas de ancoragem e *risers*, constituindo, dessa maneira, um sistema inteiramente integrado. Tais metodologias são descritas sucintamente a seguir.

2.5.3 Metodologia Desacoplada

As metodologias e critérios de projeto adotados pelas empresas de petróleo, inclusive a Petrobras, não têm tipicamente considerado a integração no projeto de sistemas de ancoragem e *risers*, sendo executados por equipes distintas de projetistas (SENRA, 2004).

A forma mais rigorosa de se efetuar um projeto integrado, considerando a contribuição dos *risers* como parte do sistema ancorado, é empregando um programa baseado em uma formulação totalmente acoplada. Tal formulação pode ser considerada em fases mais avançadas do projeto e é capaz de fornecer em uma única análise, tanto os movimentos do casco quanto uma análise estrutural detalhada do sistema de ancoragem e dos *risers*. No entanto, seria necessária a representação de todas as linhas através de uma malha de Elementos Finitos consideravelmente refinada, o que, dependendo da quantidade de linhas, pode requerer custos computacionais excessivamente elevados, comprometendo assim a viabilidade da análise. Sendo assim, é necessário estabelecer níveis diferentes de integração e complexidade em fases distintas do projeto.

Na formulação desacoplada são consideradas separadamente duas etapas distintas: a análise dos movimentos do casco da unidade flutuante, e a análise estrutural dos *risers*.

Primeiramente, realiza-se uma análise hidrodinâmica para obtenção dos movimentos do casco e uma estimativa das trações das linhas de ancoragem, sem levar em consideração o comportamento não-linear dinâmico das linhas que compõem o sistema de produção. Neste caso, as linhas são representadas, de modo simplificado, por coeficientes escalares, que representam massa, rigidez, amortecimento e parcela de carga de correnteza nas linhas, que são incorporados na equação de movimento da unidade flutuante (mesmo assim, muitas vezes somente as linhas de ancoragem são consideradas, ignorando-se os *risers*). Tais coeficientes podem ser determinados a partir de modelos analíticos simplificados (por exemplo, baseados na equação da catenária) ou calibrados a partir de modelos experimentais.

Já, na segunda etapa, os movimentos da unidade flutuante obtidos anteriormente são aplicados no topo de cada *riser*, agora representados por um modelo rigoroso de elementos finitos, para a avaliação de suas respostas estruturais.

Esta metodologia clássica surgiu para atender a sistemas flutuantes instalados em águas rasas, e introduz algumas simplificações que podem afetar consideravelmente a precisão dos resultados. Os efeitos destas simplificações se tornam mais graves com o aumento da profundidade d'água e do número de linhas do modelo, podendo comprometer a qualidade dos resultados (WICHERS & DEVLIN, 2001; HEURTIER *et al.*, 2001; KIM *et al.*, 2001; ASTRUP *et al.*, 2001, ORMBERG *et al.*, 1997; JACOB & MASETTI, 1997). Mesmo assim, essa metodologia desacoplada ainda vem sendo largamente empregada em projetos *offshore* recentes, pois o tempo computacional requerido é consideravelmente reduzido, configurando-se como sua maior vantagem.

2.5.4 Metodologia Acoplada

Com o avanço da exploração de petróleo em águas cada vez mais profundas, tem sido reconhecida a necessidade de abordagens que considerem a simulação acoplada de uma formulação completa das linhas com as equações de movimento da unidade flutuante. Tal necessidade motivou o desenvolvimento de programas baseados em uma formulação acoplada, como o programa PROSIM (JACOB, 2005) que foi o primeiro a utilizar o conceito de análise acoplada. Este programa vem sendo desenvolvido desde 1997 numa parceria entre a Petrobras e o LAMCSO – Laboratório de Métodos Computacionais e Sistemas *Offshore*, do PEC/COPPE/UFRJ.

Numa simulação acoplada, todos os efeitos não-lineares dinâmicos do sistema são incluídos implicitamente e automaticamente no esquema de análise. Assim, o equilíbrio é obtido em cada passo de tempo, garantindo um tratamento consistente entre os movimentos da unidade flutuante e a resposta estrutural das linhas, proporcionando ao profissional maior confiabilidade nos resultados (BAHIENSE, 2007).

O programa PROSIM tem sido empregado em diversos projetos da Petrobras e efetua análises estáticas e dinâmicas não-lineares de sistemas *offshore* considerando o acoplamento do casco com as linhas de ancoragem e *risers*, permitindo, assim, obter simultaneamente os movimentos da embarcação e a resposta estrutural de cada linha.

O PROSIM possui uma interface gráfica de pré e pós-processamento para gerar automaticamente os arquivos com modelos numéricos e para a visualização dos resultados. Os módulos de pré-processamento desta interface gráfica, denominada SITUA, dispõem de recursos para geração de modelos, incluindo situações de instalação e avaria de unidades ancoradas. Assim, o conjunto composto pelos sistemas de interface gráfica e programa de análise, tem sido denominado como SITUA/PROSIM.

3 MÉTODOS DE OTIMIZAÇÃO

3.1 Introdução

Problemas reais de engenharia são geralmente não-estruturados, difíceis de modelar, e complexos por natureza. Métodos convencionais de otimização encontram dificuldades como ficar preso em ótimos locais, alta dimensionalidade, e necessidade de obtenção de derivadas, o que tornam estas abordagens ineficientes para certos problemas (VIEIRA, 2009).

Métodos de otimização baseados em conceitos de evolução de indivíduos de uma população, conhecidos como Métodos Evolutivos, podem ser utilizados na busca de uma solução ótima para problemas complexos de engenharia, sendo capazes de superar as dificuldades apresentadas pelos métodos clássicos de otimização. Tais métodos utilizam uma abordagem baseada na probabilidade e na aleatoriedade para variar os parâmetros de busca.

Os Métodos Evolutivos vêm sendo desenvolvidos nos últimos tempos, mostrando uma eficiência cada vez maior na solução de problemas complexos (SIDDAL, 1982; SCHWEFEL, 1995).

Estes métodos são conhecidos como Métodos Evolutivos por serem todos baseados na evolução de um indivíduo ou de uma população de indivíduos . A forma como cada método executa esta evolução é que os difere. Uma completa análise destes métodos pode ser vista em (BÄCK *et al.*, 1997).

Nas seções seguintes serão apresentadas as idéias básicas de alguns dos métodos evolutivos mais populares: Estratégias Evolutivas, Algoritmos Genéticos, Micro Algoritmos Genéticos, Sistema Imunológico Artificial e Enxame de Partículas, sendo este último apresentado de forma mais detalhada pois o mesmo será utilizado nas análises desenvolvidas neste trabalho.

3.2 Estratégias Evolutivas (EE)

As estratégias evolutivas (EEs) foram desenvolvidas com o objetivo de solucionar problemas de otimização de parâmetros e baseiam-se na evolução de um ou mais indivíduos na direção que maximize (ou minimize) alguma característica. Foram propostas originalmente por Ingo Rechenberg e Hans-Paul Schwefel, na década de 60, que desenvolveram a chamada (1 + 1)-EE em que um pai gera um único descendente

(reprodução assexuada) e ambos competem pela sobrevivência (RECHENBERG, 1965; SCHWEFEL, 1995).

Assim, o algoritmo desenvolvido inicialmente, é composto pelas seguintes etapas:

1. *Inicialização* – A população é inicializada com um indivíduo (pai);
2. *Reprodução* – O pai da geração inicial gera, por mutação, um filho. O material genético deste filho é ligeiramente diferente do pai;
3. *Seleção* – O indivíduo mais apto, ou seja, aquele que possui o melhor valor da função objetivo sobrevive e será o pai da próxima geração.
4. O processo é repetido até que algum critério de parada seja atingido.

O modelo original de EE possui convergência lenta (DE JONG, 2006). Por isso, foram desenvolvidos posteriormente outros modelos, divididos de acordo com o mecanismo de seleção, denominados respectivamente (μ,λ)-EE e $(\mu+\lambda)$-EE, onde o símbolo μ indica o número de pais, e λ indica o número de filhos que serão produzidos em uma única geração. No primeiro, μ pais morrem e sobrevivem λ descendentes, sem competição entre os μ indivíduos já existentes com os λ novos indivíduos. No segundo modelo, sobrevivem apenas μ indivíduos selecionados entre os μ indivíduos atuais e os λ novos indivíduos gerados (os piores λ dentre todos os $\mu+\lambda$ indivíduos são descartados) (GABRIEL & DELBEM, 2008; GABRIEL, 2010; BEYER & SCHWEFEL, 2002).

A população inicial E de μ indivíduos pode ser gerada aleatoriamente ou escolhida por algum critério previamente definido. Normalmente, quando se usa $\mu > 1$ a população é gerada por processo aleatório, ou pseudo-aleatório, com distribuição uniforme para garantir que o espaço de solução seja bem representado na geração inicial (ALBRECHT, 2005).

A estratégia (μ,λ)-EE normalmente mantém uma diversidade maior de indivíduos na população, mas é possível que um indivíduo muito bom não sobreviva até o final do processo evolutivo. Já a estratégia $(\mu+\lambda)$-EE permite que uma elite de indivíduos, que são pais, domine o processo até que um indivíduo melhor seja obtido. Dessa forma, um indivíduo muito bom não é perdido e sobrevive até o final do processo evolutivo.

A representação de um indivíduo é feita através do conjunto das variáveis que definem um ponto no espaço e o operador principal é a mutação, onde um valor aleatório é adicionado a cada elemento do vetor para produzir um novo descendente. Outro operador utilizado é a recombinação intermediária (*crossover*), onde é feita uma

média aritmética dos elementos dos vetores de dois pais, elemento por elemento, para produzir um novo descendente (ABREU, 2006).

As EEs são geralmente aplicadas a problemas de otimização com valores reais, e tendem a enfatizar mutação ao invés de *crossover*. Além disso, estes algoritmos são geralmente utilizados com populações menores (de 1 a 20 indivíduos) do que os AGs. Por fim, da mesma forma que os AGs, o seu desempenho depende da configuração adequada dos seus parâmetros internos de controle. Porém, estes podem ser ajustados automaticamente através de um mecanismo de auto-adaptação ao invés do mecanismo manual geralmente utilizado nos AGs (COELHO, 2003).

Uma aplicação de Estratégias Evolutivas em sistemas de ancoragem pode ser encontrada em (SANTOS, 2000).

3.3 Algoritmos Genéticos (AG)

Os Algoritmos Genéticos (AGs) são algoritmos matemáticos inspirados na teoria de evolução das espécies de Darwin. Propostos inicialmente por HOLLAND (1975), os AGs são considerados a classe mais usual dos Algoritmos Evolutivos. Sua idéia básica é encontrar a melhor solução para um dado problema através de um processo evolutivo que seleciona a mais adaptada dentre as soluções (GOLDBERG, 1989).

Segundo a teoria de Charles Darwin, os indivíduos mais aptos possuem uma probabilidade maior de reprodução e, com isso, seus descendentes mantêm o bom material genético na espécie. Este material genético é representado pelo *cromossomo*, que constitui a identidade única de cada indivíduo.

Holland estudou a evolução natural considerando esta um processo simples e poderoso, que poderia ser adaptado para a obtenção de soluções computacionais eficientes para problemas de otimização.

O problema de otimização de uma função de *n* variáveis, $f(x): R^n \rightarrow R$, consiste em encontrar um elemento $x \in R^n$ tal que a função *f*(*x*) seja maximizada ou minimizada. Assim, nos AGs, cada indivíduo, junto com seu material genético único, representa uma solução para o problema. Cada parâmetro (m_i) é codificado por um *gene* utilizando-se de uma representação apropriada, como a representação por números reais ou uma *string* binária. A estrutura correspondente à representação de todos os parâmetros (m_1,m_2,..., m_i) descreve um *cromossomo*, que representa uma solução

individual do problema. Uma população é formada por um conjunto de soluções individuais (*cromossomos*) (VIEIRA, 2009).

A população de indivíduos é formada inicialmente de modo aleatório e, durante o processo de evolução, cada indivíduo é submetido ao processo de reprodução. A reprodução é realizada através de operadores de *crossover* (recombinação) para combinar os genes de diferentes pais e produzir seus descendentes. Os descendentes carregam características de ambos os pais, e são submetidos em seguida ao processo de *mutação*, que introduz diversidade genética à população, alterando arbitrariamente um ou mais genes do cromossomo, permitindo a introdução de novos elementos na população. Desta forma, a mutação ajuda a solucionar o problema do confinamento a mínimos locais na otimização, pois promove alterações que direcionam a pesquisa para outros locais da superfície de resposta assegurando que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será nula (CAMPOS, 2007).

Cada indivíduo é avaliado e recebe uma medida de aptidão que representa a qualidade da solução. A função que determina a aptidão de um determinado indivíduo é fortemente dependente da função objetivo $f(x)$ e é chamada de função de *fitness*. A função de *fitness* avalia cada indivíduo, medindo quantitativamente o nível de adaptação do mesmo. Quanto maior o valor do *fitness* de um cromossomo melhor é a solução codificada por este cromossomo. A busca é guiada apenas pelo valor de aptidão associado a cada indivíduo na população. Ao final da geração, através do processo de seleção, somente os indivíduos mais aptos continuam na população, os demais são descartados. Como o algoritmo genético é um processo iterativo, a seleção dos melhores indivíduos de cada geração (iteração) tem como conseqüência o aprimoramento da população, ou seja, a busca é direcionada para as regiões mais promissoras do espaço de busca.

A utilização de operadores de mutação e recombinação equilibra dois objetivos aparentemente conflitantes: o aproveitamento das melhores soluções e a exploração do espaço de busca. O processo de busca é, portanto, multidimensional, preservando soluções candidadas e provocando a troca de informação entre as soluções exploradas (MICHALEWICZ, 1996; VON ZUBEN, 2000).

O algoritmo de otimização do AG pode ser resumido como se segue:

1. *Inicialização* – Uma população inicial com n indivíduos é gerada aleatoriamente;
2. *Avaliação* – A aptidão de cada um dos n indivíduos da população é calculada utilizando a função de avaliação;

3. *Reprodução* – Os μ melhores indivíduos são selecionados para reprodução e sofrem processos de recombinação e mutação, formando λ novos indivíduos;
4. *Avaliação* – A aptidão de cada um dos λ novos indivíduos da população é calculada utilizando a função de avaliação;
5. *Seleção* – Os *n* melhores indivíduos entre os (μ+λ) indivíduos são selecionados para compor a nova população;
6. *Avaliação do critério de parada* – Os passos 3 a 5 são repetidos até que o critério de parada previamente definido seja alcançado. Esse critério pode ser um nível esperado de adequação das soluções ou um número máximo de iterações.

Os diversos métodos de seleção, formas de codificação e os tipos de recombinação podem ser encontrados em (MICHALEWICZ, 1996).

3.4 Micro Algoritmos Genéticos (Micro-AG)

Os Micro-AGs são variações de AGs desenvolvidos para encontrar soluções eficientemente utilizando populações com poucos indivíduos.

O primeiro trabalho envolvendo micro-AGs foi proposto por KRISHNAKUMAR (1989). Nesse modelo, uma população inicial bastante reduzida, N = 4, por exemplo, é gerada e evoluída até que todos os indivíduos tenham genótipos idênticos ou, pelo menos, muito similares. Com uma população tão pequena o método tenderá a convergir rapidamente para um ótimo local. Quando isto ocorre, o melhor indivíduo é mantido na população e todos os demais indivíduos são, novamente, gerados aleatoriamente. Este processo é repetido até que os critérios de parada sejam atingidos. Em geral utiliza-se alta taxa de recombinação (próxima de 1) e baixa de mutação (em muitos casos 0) (GABRIEL & DELBEM, 2008).

Micro-AGs são de grande importância em diversas aplicações, em especial em projetos onde a disponibilidade de recursos computacionais é limitada, como é o caso da área de Robótica Evolutiva (CARVALHO *et al.*, 2004), onde cada robô é um indivíduo e, como é inviável construir um grande números de robôs, a população deve ser pequena. Aplicações de tempo real também podem ser beneficiadas com a utilização de micro-AGs, uma vez que populações pequenas requerem menos tempo computacional para serem geradas (DELBEM *et al.*, 2005).

Os Micro-AGs tem demonstrado boa performance em termos de número de avaliações da função, quando comparado ao Algoritmo Genético tradicional (ALBRECHT, 2005).

3.5 Sistema Imunológico Artificial (SIA)

Os Sistemas Imunológicos Artificiais (SIA) (DASGUPTA, 1998) surgiram a partir de tentativas de modelar e aplicar princípios imunológicos no desenvolvimento de ferramentas computacionais para a solução de problemas do mundo real. Compostos por metodologias inteligentes, os Sistemas Imunológicos Artificiais são mecanismos computacionais inspirados no sistema imunológico biológico que, devido à sua capacidade de processamento de informação, já vêm sendo utilizados em diversas áreas, como reconhecimento de padrões, detecção de falhas e anomalias, segurança computacional, otimização, controle, robótica, tabulação de horários, análise de dados, aprendizagem de máquina, entre outras (DASGUPTA, 1998; BÄCK *et al.*, 2000a,b; TIMMIS, 2000; DE CASTRO, 2001; ALMEIDA, 2007; VIEIRA *et al.*, 2008a,b; VIEIRA, 2009).

O sistema imunológico é responsável pela defesa do organismo especialmente de animais vertebrados contra o ataque de antígenos (organismos externos que podem causar danos). Tal sistema classifica as moléculas do organismo como próprias e não-próprias e, para as últimas, aciona um tipo apropriado de defesa a partir dos leucócitos (DE CASTRO, 2001). A resposta do sistema imunológico pode ser uma resposta característica do sistema imune inato (resposta imediata que não exige pré-infecção) bem como uma resposta característica do sistema imune adaptativo (resposta duradoura que exige que o organismo já tenha sido infectado anteriormente por certo antígeno).

O princípio da seleção clonal estabelece que somente a célula capaz de reconhecer certo antígeno é proliferada (DE CASTRO & VON ZUBEN, 2000). Uma vez que cada célula pode conhecer apenas um número definido de invasores, aquelas que reconhecem antígenos próprios são eliminadas dentre os anticorpos antes que possam responder ao estímulo (seleção negativa), e aquela que possui maior afinidade com o antígeno é escolhida para ser clonada. Após a clonagem, os clones sofrem hipermutação somática, que é uma mutação somática feita a altas taxas, com o objetivo de melhorar a afinidade destes com o invasor.

A eficiência de um sistema imunológico é definida em termos de sua capacidade de adaptação da resposta imune em cada ataque que o organismo sofre. Também é

levado em consideração o fato de que o número de anticorpos atacando uma população de antígenos seja suficiente para eliminá-la.

O Algoritmo da Seleção Clonal (CLONALG - *Clonal Selection Algorithm*) é inspirado no Princípio da Seleção Clonal, mais especificamente na geração e proliferação de clones, na presença de antígenos e na modificação dos clones através da aplicação da hipermutação somática, e é o algoritmo baseado em Sistemas Imunológicos Artificiais mais amplamente difundido. O princípio da seleção clonal pode ser ilustrado através da Figura 25.

O algoritmo CLONALG foi inicialmente proposto na resolução de aprendizado de máquina e reconhecimento de padrões sendo, posteriormente, adaptado a problemas de otimização (DE CASTRO, 2001).

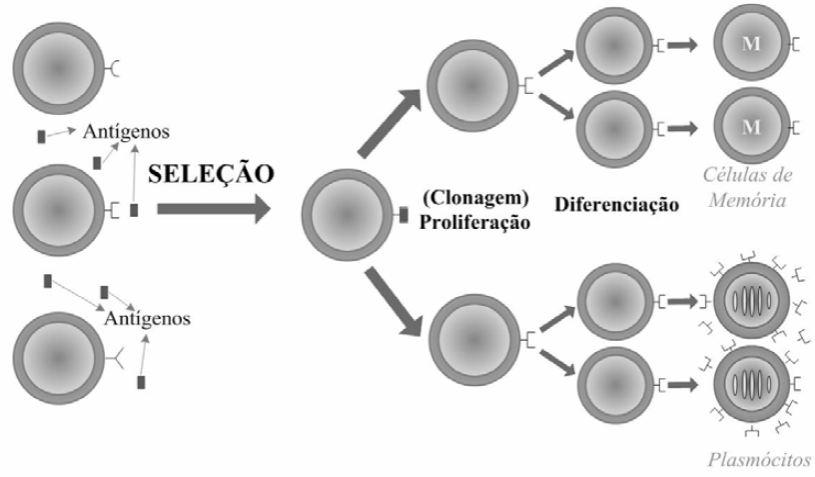

Figura 25. Princípio da seleção clonal (DE CASTRO & VON ZUBEN, 2000).

3.6 Enxame de Partículas (PSO)

O método de otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO) foi desenvolvido por KENNEDY & EBERHARDT (1995a), baseados em observações do comportamento social dos animais tais como bandos de pássaros e cardumes de peixes (Figura 26).

No método de PSO, cada indivíduo ou "partícula" de uma dada população ou "enxame" move-se através do espaço de busca em iterações sucessivas, cooperando e competindo com outras partículas. Ele "voa" lembrando da melhor posição do espaço de busca que visitou, e em direção ao melhor indivíduo de uma vizinhança topológica.

Figura 26. Movimento de bando de pássaros que inspirou a criação do PSO.

Cada partícula aprende com suas próprias experiências anteriores e com a experiência de seus vizinhos, avaliando a si própria, comparando seu desempenho com os das outras, e imitando somente aqueles indivíduos com mais sucesso do que ela. Consequentemente, os movimentos através do espaço de busca são guiados pelas melhores avaliações, com a população usualmente convergindo para uma boa solução do problema. A qualidade das soluções é medida por uma função predefinida de aptidão (*fitness*), que depende do problema.

Considerando um espaço de busca L-Dimensional, cada partícula é representada por vetores L-Dimensionais, incluindo sua posição atual no espaço de busca (x_i). Em cada iteração do algoritmo, a posição atual é medida avaliando-se a *fitness* da solução do problema. Se essa posição for melhor do que todas encontradas até então, as coordenadas são armazenadas em um outro vetor L-Dimensional p_i, que representa a melhor posição visitada previamente pela partícula. O objetivo, naturalmente, é manter-se encontrando melhores posições e atualizando p_i.

O vôo de uma partícula através do espaço de busca é expresso como um movimento a partir da posição x na iteração t com uma "velocidade" ou mudança na posição v_i:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{1}$$

Isto é, a nova posição é ajustada adicionando a x_i as coordenadas do vetor v_i, que pode efetivamente ser visto como um tamanho de passo (POLI *et al.* 2007). O algoritmo opera ajustando o vetor de velocidade v_i, que é atualizado de acordo com a equação:

$$v_i(t+1) = C_1 * rnd(p_g - x_i(t)) + C_2 * rnd(p_i - x_i(t)) + \omega * v_i(t) \tag{2}$$

Nesta expressão, $(p_g - x_i)$ e $(p_i - x_i)$ são os termos de aceleração pela distância (KENNEDY & EBERHARDT, 1995a), porque controlam a variação da velocidade medindo a distância da posição atual da partícula até a melhor posição global (ou a melhor posição visitada por todas as partículas do enxame) p_g, e até a melhor posição visitada pela partícula p_i. Nesses termos, o *rnd* representa números aleatórios amostrados a partir de uma distribuição uniforme no intervalo [0, 1].

Os coeficientes C_1 e C_2 que afetam estes termos são respectivamente os parâmetros social e cognitivo, uma vez que ajustam o balanço entre a influência social (o conhecimento do enxame inteiro) e a aprendizagem cognitiva da partícula individual.

Finalmente, a inércia no comportamento do movimento da partícula é levada em consideração pelo parâmetro ω que é o peso da inércia introduzido para ajustar o balanço entre a busca global e a busca local (SHI & EBERHARDT, 1998).

O algoritmo básico de PSO é construído então como segue:

1. *Inicialização* – A população de partículas é inicializada com posições e velocidades aleatórias em L dimensões no espaço de busca.
2. *Avaliação* – Para cada partícula, a função de aptidão desejada é avaliada em L variáveis.
3. *Atualização do melhor local* – A aptidão avaliada para a partícula na iteração atual é comparada com o melhor valor de aptidão para a partícula em todas as iterações precedentes (armazenadas em $pbest_i$). Se o valor atual é melhor do que $pbest_i$, então é atribuído a $pbest_i$ o valor atual de aptidão, e a p_i é atribuída a posição atual x_i no espaço L-dimensional.
4. *Atualização do melhor global* – A partícula na vizinhança com maior sucesso até então é identificada, atualizando p_g e *gbest*. Esta última variável armazena o valor da melhor aptidão para o enxame inteiro.
5. *Atualização os parâmetros de movimento das partículas* – A velocidade e a posição de cada partícula são atualizadas de acordo com as equações (2) e (1).
6. *Avaliação do critério de parada* – Os passos 2 a 5 são repetidos até que os critérios de parada sejam satisfeitos (geralmente uma aptidão suficientemente boa ou um número máximo de iterações).

Para reduzir a possibilidade de partículas que voam fora do espaço de busca, KENNEDY & EBERHARDT (1995b) propuseram um esquema que limita a velocidade de cada partícula a um intervalo $[-v_{max}, +v_{max}]$, com v_{max} calculada da seguinte forma:

$$v_{\max} = x_{\max} - x_{\min} \quad (3)$$

onde $x_{\max}$ e $x_{\min}$ são os limites inferior e superior do espaço de busca, respectivamente.

Muitas alterações e ajustes foram feitos no algoritmo padrão de PSO durante a década passada. Alguns resultaram em melhor desempenho geral, e alguns melhoraram o desempenho em problemas particulares. Algumas dessas variações são apresentadas nas seções a seguir.

3.6.1 Congregação Passiva e Atração Social

O algoritmo de PSO é inspirado no comportamento social com ordem espacial, como um bando de pássaros, cardumes de peixes, ou enxames de insetos. Cada um desses casos tem integridades espaço-temporais estáveis do grupo de organismos: o grupo move-se persistentemente como um todo sem perder a forma e a densidade (HE *et al.*, 2004). Para cada um desses grupos, as forças biológicas de agregação e congregação, que podem ser ativas ou passivas, são essenciais para preservar a integridade do grupo.

A força de agregação refere-se a um agrupamento dos organismos por forças físicas externas não-sociais. A agregação ativa é um agrupamento por recursos atrativos tais como o alimento, com cada membro do grupo recrutado para uma posição específica ativamente. A agregação passiva é um agrupamento por forças físicas, tais como correntes de água.

Os termos $C_1 * rnd * (p_g - x_i(t))$ e $C_2 * rnd * (p_i - x_i(t))$ da equação (2) podem ser correlacionados à agregação ativa, se p_g e p_i forem considerados recursos atrativos para os membros do grupo.

Diferente da agregação, congregação é um agrupamento por forças sociais, isto é, o recurso atrativo é o próprio grupo. A congregação passiva é uma atração de um indivíduo por outros membros do grupo onde não há nenhuma demonstração de comportamento social. A congregação ativa, também conhecida como congregação social, acontece geralmente em um grupo onde os membros são relacionados.

HE *et al.* (2004) propôs a introdução de um termo relativo à força de congregação passiva. Isso envolve uma partícula do enxame selecionada aleatoriamente e um coeficiente de congregação passiva. ALBRECHT (2005) propôs uma abordagem similar para a influência do grupo no indivíduo denominada atração social, onde o termo de

congregação passiva envolve o centro de massa do enxame, não uma partícula do enxame selecionada aleatoriamente. Esse termo novo é definido como segue:

$$C_3 * rnd * (C_m - x_i(t)) \tag{4}$$

onde C_3 é o coeficiente de congregação passiva, e C_m é o centro de massa do enxame, com a massa de cada partícula representada pelo valor de aptidão da partícula (que é sempre maior do que zero):

$$C_m = \frac{\sum_{j=1}^{N} f_j x_j}{\sum_{j=1}^{N} f_j} \tag{5}$$

Considerando esse novo termo, a equação (2) que define o vetor da velocidade no algoritmo de PSO é agora escrita como:

$$\begin{aligned} v_i(t+1) = C_1 * rnd(p_g - x_i(t)) + C_2 * rnd(p_i - x_i(t)) + \\ + C_3 * rnd(C_m - x_i(t)) + \omega * v_i(t) \end{aligned} \tag{6}$$

3.6.2 Variação Linear e Não-Linear do Coeficiente de Inércia

O algoritmo de PSO é muito eficiente no início da busca, mas pode apresentar dificuldades para convirgir para o ótimo global nas últimas iterações, desperdiçando iterações (ou tempo de processamento) quando está perto do ótimo.

O peso de inércia realiza uma tarefa importante nesse comportamento de convergência do algoritmo. Em (SHI & EBERHARDT, 1998) vários experimentos foram conduzidos para avaliar a influência desse parâmetro. Concluíram que, para valores menores do peso de inércia, o PSO se comporta como um algoritmo de busca local e, para valores maiores, o PSO apresenta uma boa exploração global, tentando sempre explorar novas áreas.

EBERHARDT & SHI (2000) propuseram que o coeficiente de inércia ω, em vez de ser constante, deveria variar linearmente. Portanto, o coeficiente de inércia é dado por:

$$\omega(t) = \omega_0 \frac{(N - t)}{N} \tag{7}$$

onde N é o número máximo de iterações, ω_0 são os pesos iniciais de inércia e t é a iteração atual.

Para melhorar o desempenho nas últimas iterações, CHATTERJEE & SIARRY (2006) propuseram uma variação não-linear do coeficiente de inércia, dada por:

$$\omega(t) = \left\{ \frac{(N-t)^n}{N^n} \right\} (\omega_{ini} - \omega_{fin}) + \omega_{fin} \tag{8}$$

onde n é o exponente de não-linearidade. Com essa formulação, a influência da inércia pode ser controlada. Os valores inicial e final do coeficiente de inércia são predeterminados, mas seu comportamento durante as iterações depende do valor de n.

Uma expressão alternativa para a variação não-linear de ω é considerada neste trabalho:

$$\omega(t) = \omega_0 \left\{ 1 - \frac{(t-1)^n}{N^n} \right\} \tag{9}$$

onde n e ω_0 são os elementos de controle da variação não-linear. Usando a equação (9), o peso de inércia ω será ω_0 no início do processo, e tenderá a zero no final.

3.6.3 Variação Linear dos Coeficientes de Agregação e Congregação

Seguindo a idéia da variação linear do peso de inércia, RATNAWEERA *et al.* (2004) propuseram a variação linear dos coeficientes de agregação (C_1 e C_2). As equações para essas variações são dadas nas equações (10) e (11).

$$C_1(t) = (C_{1\,fin} - C_{1\,ini}) \frac{t}{N} + C_{1\,ini} \tag{10}$$

$$C_2(t) = (C_{2\,fin} - C_{2\,ini}) \frac{t}{N} + C_{2\,ini} \tag{11}$$

onde $C_{1\,ini}$, $C_{1\,fin}$ são os valores inicial e final de C_1 e $C_{2\,ini}$, $C_{2\,fin}$ são os valores inicial e final do C_2, respectivamente. Essas modificações melhoraram o desempenho do algoritmo, como mostrado em (RATNAWEERA *et al.*, 2004; EBERHARDT & SHI, 1999).

A implementação do algoritmo de PSO na ferramenta *ProgOtim*, que será utilizada nos experimentos apresentados neste trabalho, incorpora as variações descritas nesta seção, incluindo a variação linear do coeficiente de congregação (C_3), dado por

$$C_3(t) = (C_{3\,fin} - C_{3\,ini}) \frac{t}{N} + C_{3\,ini} \tag{12}$$

onde $C_{3\,ini}$, $C_{3\,fin}$ são os valores inicial e final de C_3, respectivamente.

4 METAMODELOS

Os problemas de otimização de projetos de estruturas *offshore* são muito complexos. Eles exigem muitas computações, porque a maioria dos processos são simulados na base do Método de Elementos Finitos (MEF). Durante a otimização, cada iteração exige um cálculo demorado, realizado por meio do MEF, do problema considerado, o que aumenta o tempo de computação dos procedimentos de otimização.

Apesar dos avanços na capacidade dos computadores, o enorme custo computacional necessário para executar simulações complexas de engenharia torna impraticável confiar exclusivamente na simulação para fins de otimização de projeto. Para reduzir o custo, os metamodelos, igualmente conhecidos como modelos substitutos (*surrogate models*), são construídos e então usados no lugar dos modelos reais de simulação. Os metamodelos ajudam na otimização da simulação fornecendo um objetivo determinístico com tempos de execução que são geralmente muito mais curtos do que o tempo da simulação original do evento discreto.

O termo "metamodelo" foi utilizado pela primeira vez em meados da década de 70 (BLANNING, 1974). As primeiras aplicações tinham como objetivo auxiliar no cálculo de sensibilidades dos modelos de simulação, onde surgia a necessidade de se executar o metamodelo diversas vezes (BLANNING, 1974; 1975; KLEIJNEN, 1975; MICHEL & PERMUT, 1975).

Uma vertente na pesquisa em metamodelos é o estudo de erros de aproximação dos metamodelos: dado um problema, passa-se ao estudo dos metamodelos que melhor se adequam ao problema (KLEIJNEN & SARGENT, 2000). O emprego de metamodelos é estudado há muito tempo nos métodos clássicos de otimização. Por exemplo, no método Broyden–Fletcher–Goldfarb–Shanno (BFGS), que é utilizado em problemas de otimização não-linear, é construída uma aproximação da inversa da matriz hessiana, evitando um custoso processo de inversão.

Na década de 80 surgiu a primeira aplicação de metamodelos para substituir avaliações em computação evolucionária (GREFENSTETTE & FITZPATRICK, 1985). A partir de então, os metamodelos vêm sendo progressivamente estudados e atualmente são usados para:

- Auxiliar no cálculo de sensibilidades de modelos de simulação.
- Substituir simulações que demandam muitos recursos computacionais.

- Melhorar o desempenho de algoritmos iterativos de otimização mantendo fixo o custo computacional.

Atualmente, o uso de metamodelos vem encontrando mais espaço em aplicações, em geral de grande complexidade, em alguns casos intratáveis. Várias técnicas já são amplamente usadas com o mesmo objetivo, como o uso de polinômios, técnicas de agrupamento de dados, uso de modelos estatísticos, redes neurais e muitos outros (FONSECA, 2009).

Neste capítulo serão apresentados, de forma suscinta, a teoria básica de metamodelos, os tipos de estratégias de otimização de projeto baseadas em metamodelagem e algumas das principais técnicas de metamodelagem: Superfície de Resposta, Kriging, Spline de Regressão Adaptativa Multivariada, Funções de Base Radial e Redes Neurais Artificiais.

O método de Redes Neurais Artificiais foi escolhido para o desenvolvimento deste trabalho e, portanto, a teoria de RNAs será descrita em maiores detalhes no capítulo 5.

4.1 Introdução

O projeto/reprojeto de sistemas é um processo complexo em que os modelos são usados para fazer decisões sobre mudanças em sistemas existentes ou propostos. O objetivo do processo de projeto é projetar um sistema que encontre ou exceda determinadas medidas de desempenho sem violar nenhuma restrição. A modelagem de simulação é uma das ferramentas mais populares para o projeto e a análise de sistemas complexos. Esta popularidade se deve à sua flexibilidade, à sua habilidade de modelar sistemas com maior exatidão e à sua habilidade de modelar o comportamento dinâmico do sistema no tempo. Com a modelagem de simulação, entretanto, a relação entre os parâmetros de projeto e as medidas de desempenho não é conhecida explicitamente. Consequentemente, o projeto de sistemas usando a simulação torna-se um processo de tentativa e erro no qual um conjunto de parâmetros de projeto é usado no modelo de simulação para prever um conjunto de medidas de desempenho. Se as medidas de desempenho são aceitáveis, um bom projeto foi identificado; se não, o processo é repetido até que um conjunto satisfatório de medidas de desempenho seja obtido. Por causa da natureza iterativa deste procedimento, este processo pode ser demorado e caro (NASEREDDIN & MOLLAGHASEMI, 2005).

A maioria dos sistemas podem ser vistos como uma função de transformação entre as variáveis de entrada e as variáveis de saída. O modelo de simulação é construído para aproximar tanto quanto possível a realidade. Entretanto, para sistemas muito complexos, o modelo de simulação será grande e difícil de compreender. Restrições, tais como o custo e a dificuldade do desenvolvimento do modelo, podem impedir a construção de múltiplos protótipos do sistema real (SONG *et al.*, 2008).

Para superar as limitações dos modelos de simulação, pesquisadores desenvolveram os metamodelos, que são aproximações do modelo de simulação e computacionalmente mais eficientes (GIUNTA *et al.*, 1998). A finalidade principal do metamodelo é reproduzir com precisão a simulação e reduzir o custo, o tempo de computação e a quantidade de esforços exigidos durante uma análise da simulação.

Muitas definições de metamodelo estão disponíveis. BARTON (1992) define o metamodelo como um modelo do modelo de simulação e exibe a relação fundamental entre a entrada e a saída. Depois desta definição, muitos teoremas famosos podem ser vistos como metamodelos. Por exemplo, a Lei de Little $L = \lambda W$ é um metamodelo de uma simulação de fila (BARTON, 1992). SANTOS & SANTOS (2007) definem o metamodelo como abstrações do modelo de simulação que exibem o relacionamento entrada-saída do sistema através de uma simples expressão matemática. O metamodelo fornece uma maneira analítica de estudar o comportamento de um sistema complexo. O processo de construção de um metamodelo é chamado de metamodelagem.

No desenvolvimento de metamodelos de simulação, duas abordagens têm sido utilizadas: a metamodelagem de simulação direta e a metamodelagem de simulação reversa. Ao construir o metamodelo utilizando aproximação direta, as entradas da simulação (parâmetros de projeto) são usadas como entradas para o metamodelo e as saídas da simulação (medidas de desempenho) são usadas como saídas desejadas para o metamodelo. O metamodelo direto é geralmente aplicado em problemas de predição ou otimização. Quando aplicado em otimização, geralmente é usado em conjunto com uma técnica de otimização (por exemplo GA, PSO e *simulated annealing*). Ao construir um metamodelo de simulação reversa, as saídas da simulação (medidas de desempenho) são usadas como entradas para o metamodelo e as entradas da simulação (parâmetros de projeto) são usadas como saídas desejadas para o metamodelo. O metamodelo de simulação reversa é geralmente usado para conseguir níveis específicos de medidas de desempenho. A vantagem de se usar um metamodelo de simulação reversa como um instrumento de apoio de decisão é que o processo já não é iterativo. O responsável pelas

decisões insere o nível exigido de medidas de desempenho e as saídas do metamodelo serão os parâmetros necessários para conseguir as medidas desejadas (NASEREDDIN & MOLLAGHASEMI, 2005).

4.2 Estratégias de Otimização de Projeto Baseadas em Metamodelagem

Três tipos diferentes de estratégias de otimização de projeto baseadas em metamodelagem podem ser encontrados na literatura, como ilustrado na Figura 27 (WANG & SHAN, 2007). A primeira estratégia (Figura 27a) é a aproximação sequencial tradicional, isto é, um metamodelo global é ajustado e então usado como um substituto da função custosa. Esta abordagem usa um número relativamente grande de pontos de amostra no princípio. Pode ou não incluir um estágio sistemático de validação do modelo. Se sim, o método de validação pode ter que ser o *cross-validation*. Esta abordagem é geralmente encontrada na literatura (BERNARDO, *et al.*, 1992; GU, 2001; MYERS & MONTGOMERY, 1995; GOMES, 2007). Esta primeira abordagem será utilizada nas análises apresentadas neste trabalho.

A segunda abordagem (Figura 27b) envolve a validação e/ou otimização iterativa, criando, a cada passo, novas amostras para remodelar o metamodelo. Em (DENNIS & TORCZON, 1996), as amostras foram geradas iterativamente para atualizar a aproximação e manter a exatidão do modelo. OSIO & AMON (1996) desenvolveram uma estratégia de *kriging* multi-estágios para atualizar sequencialmente e melhorar a precisão das aproximações substitutas enquanto pontos de amostra adicionais eram obtidos. SCHONLAU *et al.* (1998) descreveu um algoritmo sequencial para balancear buscas locais e globais usando aproximações durante uma otimização com restrições. WANG *et al.* (2001) desenvolveram uma série de amostragems adaptativas e métodos de metamodelagem para otimização, nos quais a otimização e a validação são usadas para formar um novo conjunto de amostras (WANG *et al.*, 2001). FONSECA (2009) apresenta uma metodologia baseada em metamodelos adaptativos associados a algoritmos genéticos aplicado-a na otimização de problemas mono e multi-objetivo.

A terceira abordagem é mais recente e gera diretamente novos pontos de amostra próximos do ótimo com a orientação de um metamodelo (WANG *et al.*, 2004; SHAN & WANG, 2005a,b). Diferente das duas primeiras abordagens, o metamodelo não é usado como um substituto em um processo típico de otimização. A otimização é realizada apenas pela amostragem adaptativa e nenhum processo formal de otimização é

utilizado. O metamodelo é usado como um guia para a amostragem adaptativa e, consequentemente, a necessidade de exatidão do modelo é reduzida. Este método ainda precisa ser testado para problemas de alta dimensão.

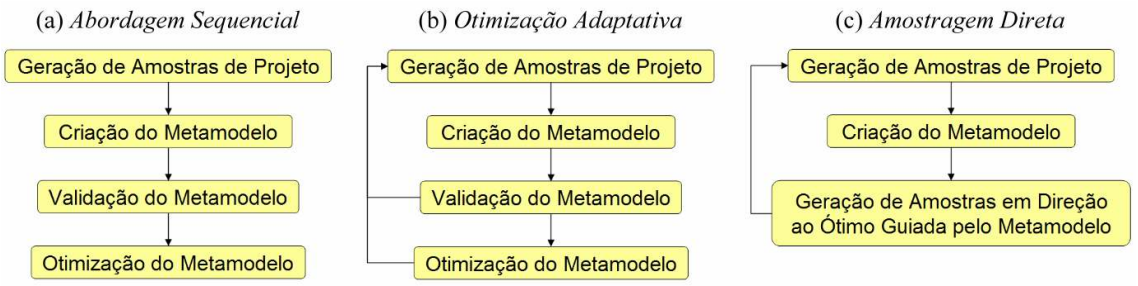

Figura 27. Estratégias de otimização de projeto baseadas em Metamodelagem.

4.3 Técnicas de Metamodelagem

Técnicas de metamodelagem têm sido amplamente utilizadas em muitas aplicações de engenharia para a otimização de projeto e para reduzir o tempo de solução da análise que envolve programas de simulação computacionalmente caros e demorados (BARTON, 1998). BARTHELEMY & HAFTKA (1993) apresentam uma revisão de aplicações de metamodelagem em otimização estrutural. SOBIESZCZANSKI-SOBIESKI & HAFTKA (1997) apresentam aplicações de metamodelagem na otimização de projeto multidisciplinar. Um estudo comparativo de várias técnicas de metamodelagem foi desenvolvido por JIN *et al.* (2000) usando critérios de modelagem diferentes em uma variedade de problemas de teste.

Os metamodelos podem ser classificados em técnicas paramétricas e não-paramétricas (HUSSAIN *et al.*, 2002). As técnicas paramétricas aproximam funções a-priori, sem nenhum conhecimento prévio sobre os dados. Estas funções são usadas para ajustarem a resposta observada por meio do ajuste dos coeficientes das funções escolhidas. Como exemplos de técnicas paramétricas temos modelos polinomiais, modelos lineares gerais, modelos não-lineares, e modelos de Taguchi. As técnicas não-paramétricas não possuem um conjunto a-priori de funções; ao invés disso, usam um método a-priori para construir uma aproximação da função baseada em respostas observadas. Como exemplos de técnicas não-paramétricas temos a função de base radial, e modelos baseados em *splines* (KINGRE, 2004).

Esta seção apresenta técnicas de metamodelagem para simulações, tais como metodologias de Superfície de Resposta, Kriging, Spline de Regressão Adaptativa Multivariada, Funções de Base Radial e Redes Neurais Artificiais.

4.3.1 Superfície de Resposta

A metamodelagem por Superfície de Resposta, formada por uma coleção de técnicas estatísticas e matemáticas de modelagem, é a técnica mais popular de metamodelagem. Tem sido usada eficazmente para a construção de aproximações simples e rápidas de códigos complexos de computador durante os últimos trinta anos (MAHDI & MANSOOREH, 1999).

Inicialmente o objetivo da modelagem de superfície de resposta era analisar os resultados de experiências físicas e gerar modelos baseados empiricamente dos resultados observados (valores de resposta).

O método de Superfície de Resposta é representado pela seguinte equação:

$$y(x) = f(x) + \varepsilon \tag{13}$$

onde $y(x)$ é a função de interesse não identificada, $f(x)$ é uma função polinomial conhecida de x, e ε é um erro aleatório que assume-se ser normalmente distribuído com média zero e variância σ^2. O erro individual ε_n em cada observação também é assumido ser independente e identicamente distribuído (KINGRE, 2004).

Esta abordagem geralmente é melhor ajustada usando-se polinômios de primeira ou segunda ordem de y (saída). No caso de polinômios de ordens maiores, podem ocorrer instabilidades (BARTON, 1992), ou pode ser demasiado complicado fazer a amostragem de dados adequada para aproximar todos os coeficientes na equação do polinômio. Sendo assim, na equação (13), a função polinomial $f(x)$ é um polinômio de baixa ordem, e é usado para aproximar $y(x)$, que assume-se ser linear, vide equação (14), ou quadrático, vide equação (15) (MYERS & MONTGOMERY, 1995).

$$\hat{y} = \beta_0 + \sum_{n=1}^{k} \beta_n x_n \tag{14}$$

$$\hat{y} = \beta_0 + \sum_{n=1}^{k} \beta_n x_m + \sum_{n=1}^{k} \beta_{nn} x_n{}^2 + \sum_{n} \sum_{m} \beta_{nm} x_n x_m \tag{15}$$

O método de regressão dos mínimos quadrados (*Least Square Regression* - LSR) é usado para determinar os coeficientes β_0, β_n, β_{nn} e β_{nm}. O método de LSR minimiza a soma de desvio quadrático dos valores previstos $\hat{y}(x)$ a partir dos valores reais $y(x)$. Os coeficientes β podem ser encontrados usando regressão dos mínimos quadrados dada por:

$$\beta = \left[X'X\right]^{-1}X'y \tag{16}$$

Na equação 16, X é a matriz de entrada dos pontos de dados de teste e y é um vetor coluna que contém a medida de desempenho em cada ponto de teste (MAHDI & MANSOOREH, 1999; MYERS & MONTGOMERY, 2002).

4.3.2 Kriging

Kriging, também chamada de Krigagem, é uma metodologia baseada em interpolação de dados (SIMPSON *et al.*, 1998). A validade do modelo de Kriging é independente do erro aleatório. Pode ser melhor ajustada para aplicações que envolvem experimentos de computador porque pode "respeitar os dados" ou fornecer uma interpolação exata ou "suavizar os dados" (SIMPSON *et al.*, 1998; CRESSIE *et al.*, 1993).

O uso de Kriging é restringido a uma região muito pequena porque envolve a inversão e a multiplicação de um grande número de matrizes. Uma comparação de Kriging com Regressão Polinomial pode ser vista em (SIMPSON *et al.*, 1998).

O modelo de Kriging é baseado na equação (17); ele postula uma combinação da função global ($f(x)$) e residual ($Z(x)$) (JIN *et al.*, 2000):

$$y(x) = f(x) + Z(x) \tag{17}$$

Na equação (17), $y(x)$ é uma função de interesse desconhecida, $f(x)$ é conhecido e é geralmente uma função polinomial de x, e $Z(x)$ é assumido para ser uma realização de um processo estocástico com média zero, variância σ^2 e tendo uma função de correlação espacial (JIN *et al.*, 2000) na matriz de co-variância de $Z(x)$ dada por:

$$Cov\left[Z(x_n), Z(x_m)\right] = \sigma^2 \boldsymbol{R}\left[R(x_n, x_m)\right] \tag{18}$$

onde $\boldsymbol{R}$ é uma matriz de correlação e $R(x_n, x_m)$ é uma função de correlação. Um conjunto de funções de correlação poder ser encontrado em (SIMPSON *et al.*, 1998). A

habilidade de usar uma larga escala de funções de correlação faz o método extremamente flexível. Apesar de muitas vantagens, tais como ser um algoritmo por etapas para selecionar os fatores vitais e selecionar fatores da entrada para construir um modelo predictor (JIN *et al.*, 2000), aplicações de Kriging são limitadas por causa da computação demorada da máxima verossimilhança (θ) que é usada para ajustar o modelo no problema de otimização *K*-dimensional. Um conjunto de problemas resolvidos usando o método de Kriging pode ser visto em (JIN *et al.*, 2000).

4.3.3 Spline de Regressão Adaptativa Multivariada

A metamodelagem por Spline de Regressão Adaptativa Multivariada (*Multivariate Adaptive Regression Spline Metamodeling* - MARS) é um método estatístico novo apresentado por (FRIEDMAN, 1991). Este método tenta aproximar relacionamentos complexos por uma série de regressões lineares em diferentes intervalos dos limites da variável independente ou das subregiões do espaço da variável independente. É muito flexível uma vez que pode se adaptar a qualquer forma funcional. MARS constrói uma relação a partir de um grupo de coeficientes (funções de base) que são inteiramente determinados pelos dados da regressão. Para aproximação, MARS usa uma função *two-sided* truncada, também conhecida como abordagem iterativa *forward-backward*.

Um modelo de MARS pode ser postulado como:

$$\hat{y} = \sum_{n=1}^{N} a_n B_{kn}(x_{v(k,n)}) \tag{19}$$

Na equação (19), $\hat{y}$ é a variável dependente (do resultado), a_n é o coeficiente de expansão, e B_{kn} são as função de base, definidas como:

$$B_{kn}(x_{v(k,n)}) = \prod_{k=1}^{K} h_{kn} \tag{20}$$

onde x é a variável do preditor, k é a ordem da interação e n representa o número de termos. Para a primeira ordem (k = 1) de interação, o modelo é aditivo, e para maiores que um (k > 1), o modelo é interativo par-a-par (FRIEDMAN, 1991). MARS busca o espaço de projeto inteiro, e durante esta busca, um crescente número de funções de base são adicionados ao modelo. MARS determina automaticamente as variáveis

independentes mais importantes assim como a mais significativa interação entre as variáveis independentes e dependentes (KINGRE, 2004).

MARS é relativamente novo comparado às outras técnicas. A exatidão e a redução do custo da computação são as vantagens principais deste método.

4.3.4 Funções de Base Radial

A técnica de metamodelagem por Função de Base Radial (*Radial Basis Function* - RBF) é baseada em interpolação. Ela fornece uma outra abordagem para metamodelagem multivariada. Em uma comparação experimental descobriu-se que o RBF é melhor do que as metodologias de Superfície de Resposta, Spline de Regressão Adaptativa Multivariada e Kriging (FRANKE *el al.*, 1982),. Uma vez que é um método de interpolação, assim como o Kriging, sua abordagem direta para metamodelagem de simulação também é restringida a uma região pequena.

O modelo de RBF é construído como uma combinação linear de *K* funções de Base Radial com *K* centros. As funções de base radial baseam-se na métrica de distância Euclideana para aproximar a saída (medida de desempenho). O desenvolvimento original feito por HARDY (1971), introduziu, entre outras, funções de base "multiquadráticas" simples:

$$\hat{y} = \sum_{j=1}^{K} \beta_j \left\| x - x_{ej} \right\| \tag{21}$$

onde β_j é o peso ou o coeficiente associado às funções de base radial que são centradas em x_{ej}, e x_{ej} é a entrada experimental. $\hat{y}$ e β_j no metamodelo, dependem da posição da entrada observada x_{ej}. Os coeficientes β_j são encontrados substituindo o lado esquerdo da equação (21) por o $f(x_j)$, $i = 1,\ldots, n$, e resolvendo o sistema linear resultante.

A função de base "multiquardrática" fornece um bom resultado para modelos bivariados e de ordem mais elevada. As funções de base radial são também relacionadas a uma outra classe de funções *spline*. As chamadas *thin-plate splines* têm funções de base radial de $\left\| x - x_j \right\|^2 \log \left\| x - x_j \right\|$; que também fornece um ajuste melhor aos modelos bivariados a entrada dispersas. Apesar de faltar uma aplicação direta à modelagem de simulação, a RBF fornece as melhores soluções para problemas com pequenas e

escassas amostras de teste. Em (JIN *el al.*, 2000) foi concluído que a RBF obtém melhores resultados quando a exatidão média e a robustez são consideradas.

4.3.5 Metamodelagem por Redes Neurais Artificiais

A Rede Neural Artificial, assim como a metodologia de Superfície de Resposta, também é uma técnica de metamodelagem extensamente aplicada para gerar aproximações de códigos de computador complexos (MAHDI & MANSOOREH, 1999).

As RNAs têm recebido cada vez mais atenção como ferramentas de metamodelagem. As redes neurais *multi-layer perceptron* oferecem a capacidade universal de aproximação de funções baseadas completamente em dados próprios, isto é são os modelos puramente empíricos que podem, teóricamente, imitar todo o relacionamento a qualquer grau de precisão. Para o desenvolvimento de um metamodelo, a habilidade de modelar universalmente qualquer relacionamento é uma vantagem tremenda porque remove o potencial para erro oriundo da pré-seleção da forma funcional. As redes neurais não são sensíveis aos desvios das suposições tradicionais do modelo estatístico; erro de variância constante, distribuição Gaussiana dos erros, nenhuma multicolinearidade de variáveis independentes, e nenhuma autocorrelação. Adicionalmente, redes *perceptron* inteiramente conectadas são modelos globais, de forma que uma única rede neural poderia ser desenvolvida para modelar a escala inteira do interesse. Uma discussão da visão geral dos usos potenciais de redes neurais para a modelagem da simulação foi apresentada por (PADGETT & ROPPEL, 1992). FISHWICK (1989) publicou uma tentativa de usar uma rede neural como uma aproximação de uma simulação determinística. Usar uma rede neural para modelar simulações era o assunto de trabalhos de (PIERREVAL, 1992) e (PIERREVAL & HUNTSINGER, 1992). BADIRU & SIEGER (1993) usaram com sucesso uma rede neural para prever os valores futuros do fluxo de caixa para a estimação do custo usando o investimento inicial e a taxa de juros por período de tempo como variáveis aleatórias uniformes.

Similarmente aos outros modelos, a RNA também depende das entradas observadas. A exatidão e a precisão da RNA são baseadas na qualidade do conjunto de dados usado na modelagem. Similar ao modelo de regressão polinomial de alta ordem, a precisão e a exatidão do modelo diminuem quando são usados conjuntos de dados pequenos na modelagem (KINGRE, 2004).

5 REDES NEURAIS ARTIFICIAIS

5.1 Introdução

O paradigma das Redes Neurais Artificiais (RNAs) surgiu em conseqüência da busca por conhecimento a respeito do funcionamento do cérebro humano. Dessa forma, houve um grande interesse em pesquisar o papel do funcionamento dos neurônios, o qual motivou a construção de modelos computacionais que pudessem auxiliar na elucidação de aspectos neurobiológicos envolvidos em diversas atividades cognitivas. As Redes Neurais Artificiais encontram-se entre estes modelos e refletem o comportamento das redes neurais biológicas (grupos de neurônios funcionalmente interconectados). A base da interconexão das RNAs são os elementos matemáticos denominados neurônios artificiais ou simplesmente neurônios.

5.2 Histórico

A história das Redes Neurais Artificiais (RNAs) começou no início da década de 40 quando o primeiro modelo artificial de um neurônio foi apresentado pelo neurofisiologista Warren McCulloch e pelo matemático Walter Pitts em 1943 (MCCULLOCH & PITTS, 1943). Eles apresentaram uma discussão sofisticada sobre redes lógicas de nodos (neurônios) e idéias inovadoras sobre máquinas de estados finitos, elementos de decisão de limiar lineares e representações lógicas de várias formas de comportamento e memória.

Embora tenha introduzido o modelo artificial de um neurônio, o trabalho de McCulloch e Pitts não apresentou técnicas de aprendizado. Em 1949, Donald Hebb desenvolveu a primeira regra de aprendizado para redes neurais artificiais, chamada de *Regra de Hebb* (HEBB, 1949). Hebb sugeriu que a alteração na eficiência "sináptica" é a base do aprendizado, através do seguinte postulado: *"Quando uma célula A está suficientemente próxima para excitar uma célula B e repetida ou persistentemente toma parte no disparo desta, algum processo de crescimento ou mudança metabólica ocorre em uma ou ambas as células de modo que a eficiência de A em excitar B é aumentada."*. Então, segundo Hebb, se duas células (neurônios) são excitadas simultaneamente, então o peso da conexão entre elas deve aumentar.

O comportamento de aprendizado de um neurônio artificial foi tratado mais extensivamente por Frank Rosenblatt em 1958, através da introdução do *Perceptron*

(ROSENBLATT, 1958), que é a estrutura mais simples de Redes Neurais Artificiais que se pode construir. Basicamente, o *Perceptron* é uma estrutura com apenas um neurônio e pesos ajustáveis sendo capaz de classificar padrões linearmente separáveis (isto é, situados em lados opostos de um hiperplano). A saída de uma rede do tipo *Perceptron* tem valor zero ou um, conforme a classificação dada ao valor calculado (igual a 1, se o valor calculado pela rede for maior ou igual a zero; e igual a zero se o valor for menor que zero). A regra de aprendizado do *Perceptron* utiliza uma atualização iterativa dos pesos mais "poderosa" do que a *Regra de Hebb*.

Similarmente ao *Perceptron*, WIDROW & HOFF (1960) criou o *Adaline* (*Adaptive Linear Neuron*), um elemento processador linear cuja saída é a soma ponderada das entradas pelos seus respectivos pesos. Os elementos processadores também são binários, porém, variam no intervalo (-1, 1). Mais tarde foi criado o *Madaline* (*Multilayer Adaline*), uma arquitetura com duas camadas de neurônios (WIDROW & LEHR, 1990). A maior contribuição do algoritmo *Adaline* foi a invenção da *Regra Delta* ou *Método do Gradiente* para treinamento das redes, que se baseava num processo de iteração local para obtenção de um ponto mínimo, utilizando um exemplo do conjunto de treinamento por vez.

O interesse em pesquisas na área de RNAs diminuiu no final da década de 60 quando Marvin Minsky e Seymour Papert demostraram que o neurônio artificial, quando considerado isoladamente (rede neural em camada simples ou *Perceptron*), seria incapaz de mapear problemas não-linearmente separáveis, como, por exemplo, o problema XOR (ou-exclusivo). MINSKY & PAPERT (1969) também colocaram em dúvida que um algoritmo de aprendizagem pudesse ser encontrado para redes neurais multicamadas.

Devido à visão pessimista de Minsky e Papert, as pesquisas sobre redes neurais foram praticamente esquecidas durante a década de 70. No entanto, alguns pesquisadores continuaram a publicar seus trabalhos como Teuvo Kohonen e James Anderson, que divulgaram suas pesquisas em *Redes Neurais de Memória Associativa* (KOHONEN, 1972; ANDERSON, 1972).

No início dos anos 80, HOPFIELD (1982) importou da física o conceito de função de energia para explicar o comportamento de redes recorrentes com conexões simétricas. Esta classe de redes neurais passou a ser denominada de *Redes de Hopfield* (FONTOVA, 2003). Em 1982 também surgiu o modelo de mapas auto-ajustáveis desenvolvido por KOHONEN (1982). Em 1985, Gail Carpenter e Stephen Grossberg

desenvolveram a teoria de redes neurais auto-organizadoras chamada de *Teoria Adaptativa de Ressonância* (*Adaptive Resonance Theory* – ART) (CARPENTER & GROSSBERG, 1985).

Rumelhart e McClelland, em 1986, introduziram uma solução poderosa para o treino de uma rede multicamadas que ficou conhecida como o algoritmo de *Retropropagação* (*Backpropagation*) (RUMELHART & MCCLELLAND, 1986). Na verdade, o método de aprendizado por *Retropropagação* também foi descoberto independentemente por PARKER (1985) e LE CUN (1986) antes de se tornar mundialmente conhecido. Depois, descobriu-se que o mesmo algoritmo havia sido descrito por Werbos em 1974 (WERBOS, 1974). Parecido com seu antecessor, o *Perceptron*, difere deste por permitir a utilização de mais de duas camadas de neurônios sendo, por isso, conhecido também como *Perceptron* multicamadas (*Multilayer Perceptron* – MLP). Neste modelo, o padrão apresentado à camada de entrada é propagado até a camada de saída (*forward-propagation*), onde é calculado o erro entre a saída real e a saída desejada de cada neurônio da camada de saída. Este erro é propagado de volta (*backward*) através dos respectivos pesos das conexões (CUNHA, 2001). O algoritmo de *Retropropagação* resolveu diversos problemas pendentes no tocante à estimação de pesos de redes neurais e hoje é um dos métodos mais utilizados para treinamento de *Perceptrons* multicamadas. Desde então, muitos trabalhos foram desenvolvidos para melhorar o algoritmo de *Retropropagação*, tais como evitar *overfitting* dos dados, acelerar o algoritmo e evitar mínimos locais (HAYKIN, 1999; FINE, 1999).

Em 1987 surgiram as *Redes Competitivas*, também chamadas de *Redes de Hamming*, que são classificadores de máxima verossimilhança que podem ser usados para determinar qual dentre vários exemplos é mais semelhante a uma entrada (LIPPMANN, 1987; DARPA, 1988).

Em 1988, Specht construiu a *Rede Neural Probabilística* (*Probabilistic Neural Nets* – PNN) usando idéias de teoria de probabilidades tais como a classificação Bayesiana e os estimadores de probabilidade de funções de densidade para formar uma rede neural para classificação de padrões (SPECHT, 1988). Também em 1988, Linkser descreveu um novo princípio para a auto-organização em uma rede perceptiva (LINKSER, 1988), que concebe o princípio para preservar o máximo de informação sobre os padrões de atividade de entrada, sujeito a limitações como as conexões sinápticas e o intervalo dinâmico das sinapses. Ainda em 1988, Broomhead e Lowe

descreveram um procedimento de criação de redes neurais *feedforward* usando funções de base radial (*Radial Basis Function* - RBF), que se tornou uma alternativa para o uso de perceptrons multicamadas (BROOMHEAD & LOWE, 1988).

Na década de 90, muitos trabalhos foram desenvolvidos abordando uma perspectiva estatística para redes neurais (CHENG & TITTERINGTON, 1994; WARNER & MISRA, 1996), não somente usando redes neurais *feedforward*, mas também usando redes recorrentes, por exemplo, o algoritmo de *Aprendizagem Recorrente em Tempo Real* (*Real Time Recurrent Learning* – RTRL) (WILLIAMS & PENG, 1990).

Em 1995, Vapnik inventou uma classe de redes de aprendizagem supervisionada poderosa do ponto de vista computacional, chamada de *Máquinas de Vetor de Suporte* (*Support Vector Machines* - SVMs), para ser utilizada em reconhecimento de padrões, regressão e problemas de estimação de densidade (VAPNIK, 1998).

Em 1997 surgiram os *Sistemas Híbridos* como, por exemplo, as redes neurais nebulosas, os sistemas híbridos neuro-evolutivos (HARRALD & KAMSTRA, 1997) e os sistemas híbridos neuro-nebulosos-evolutivos (KHOSLA & DILLON, 1997), através da associação de modelos de redes neurais artificiais com sistemas nebulosos e algoritmos evolutivos.

Atualmente, encontram-se muitos estudos sobre Redes Neurais Artificiais na literatura e o crescente número de artigos publicados confirmam o sucesso deste modelo. Muitos trabalhos foram desenvolvidos para melhorar as técnicas de RNAs através de modificações que levam a um ponto ótimo entre o tempo de treinamento e a uma generalização eficiente. Além disso, as RNAs têm sido aplicadas em problemas de reconhecimento e classificação de padrões, *clustering*, previsão de séries temporais, aproximação de funções, predição, otimização, sistemas especialistas, processamento de sinais (imagens, sensores, voz, caracteres, visão, compressão de dados, filtragem de sinais), telecomunicações, manufatura, monitoramento de processos e robótica (WIDROW *et al.*, 1994).

5.3 O Neurônio Biológico

Os neurônios são células presentes no sistema nervoso responsáveis pela recepção, transmissão e processamento dos estímulos que chegam ao organismo ou partem dele. Individualmente, os neurônios realizam operações relativamente simples,

porém a riqueza das conexões entre estes tipos de células proporciona a enorme diversidade de tarefas realizadas pelo sistema nervoso.

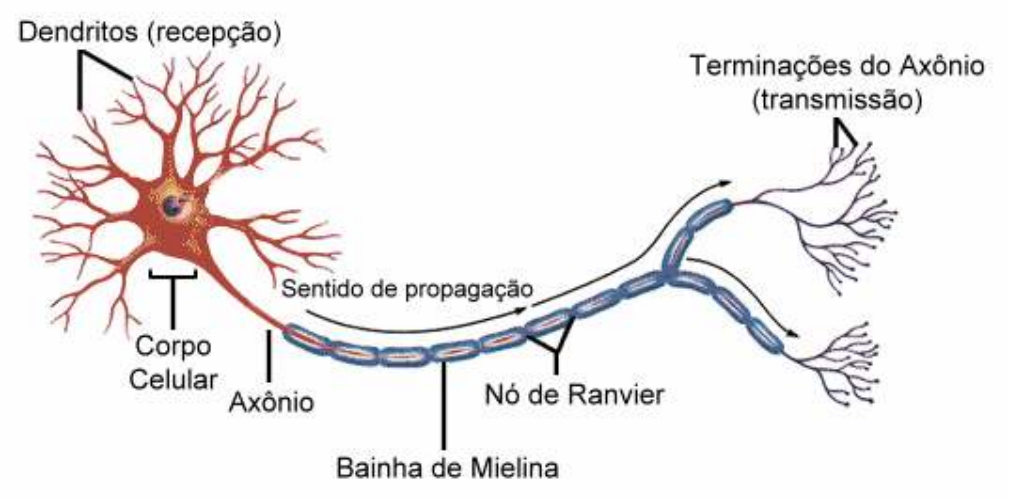

Figura 28. Neurônio Biológico.

Os neurônios biológicos são formados basicamente por três partes anatomicamente distintas: um conjunto de fibras de entrada (dendritos), o corpo da célula (soma) e uma fibra de saída (axônio). Uma configuração simplificada do neurônio biológico pode ser vista na Figura 28. O axônio se divide em diferentes terminações que se conectam aos dendritos de outros neurônios. Um neurônio pode receber mais de 10.000 entradas de outros neurônios. As estruturas em forma de bulbo onde as fibras se conectam são chamadas de sinapses. Os pulsos elétricos gerados pelos neurônios, conhecidos como impulsos nervosos, são transmitidos através do axônio até as sinapses. Quando um impulso nervoso é transmitido pela sinapse para outro neurônio, pode ocorrer a excitação ou a inibição desse sinal recebido. As sinapses desempenham um importante papel porque sua eficiência na transmissão de impulsos nervosos do axônio para os dendritos de outros neurônios pode ser alterada dependendo do benefício obtido a partir dessa alteração (VEELENTURF, 1995). A capacidade de aprendizado do ser humano é provavelmente incorporada à facilidade de mudar a eficiência de transmissão das sinapses (HEBB,1949).

5.4 O Neurônio Artificial

O primeiro modelo matemático do neurônio artificial, desenvolvido na tentativa de representar o comportamento do neurônio biológico, foi proposto por MCCULLOCH & PITTS (1943), inspirados nas propriedades eletrofisiológicas do

neurônio biológico e também nos resultados de Alan Turing e John Von Neumann, que indicavam que a natureza da inteligência humana era essencialmente booleana (TURING, 1936; VON NEUMANN, 1958).

Em um modelo artificial simplificado de um neurônio (Figura 29), também conhecido como neurônio de McCulloch-Pitts, a eficiência da transmissão das sinapses é traduzida para um número real w_i pelo qual uma entrada x_i é multiplicada antes de entrar na célula do neurônio. O número w_i é chamado de peso da entrada x_i. A ausência ou a presença de impulsos nervosos em um neurônio biológico é modelada por uma variável x_i que respectivamente pode ter o valor 0 ou 1. Nesse caso diremos que temos um neurônio artificial binário representando o comportamento "0 ou 1" de um neurônio biológico.

Na realidade, os neurônios podem disparar cerca de 100 pulsos por segundo. Podemos modelar essa mudança gradual na freqüência do pulso de treinamento por uma variável x_i que pode ter qualquer valor entre 0 e 1. Nesse caso, temos um neurônio artificial contínuo.

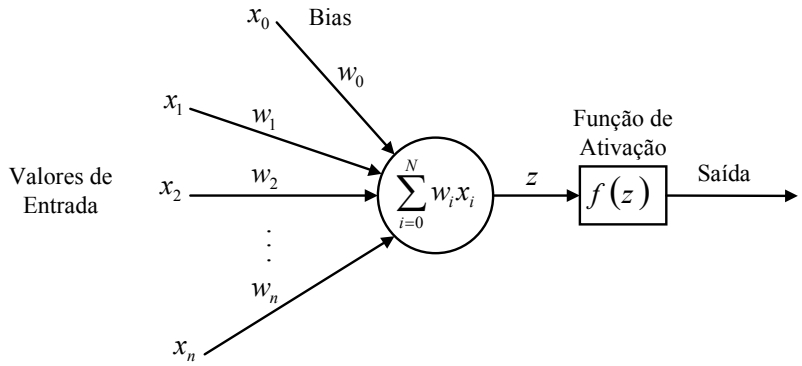

Figura 29. Neurônio Artificial de McCulloch-Pitts.

Em quase todos os modelos artificiais de neurônios, todas as entradas x_i são ponderadas de acordo com a eficiência da transmissão sináptica e são somadas obtendo um número z. Portanto, temos que o valor de z é dado pela equação:

$$z = \sum_{i=1}^{N} x_i w_i \tag{22}$$

onde i é o índice da entrada e N é o número total de entradas.

Essa entrada ponderada z determina de uma forma ou de outra o valor de saída y do neurônio artificial. Em um neurônio binário, a saída y será 1 (o neurônio está

excitado) se as entradas ponderadas excedem algum valor limiar T (*threshold*), e será zero (o neurônio está inibido) se a entrada ponderada está abaixo desse limite. Em um neurônio artificial contínuo, a saída y pode ser alguma função monótona crescente f da entrada ponderada z (a freqüência dos pulsos no axônio de saída muda gradualmente dependendo do valor de z). Tal função é chamada de função de ativação do neurônio. Assim, a saída do neurônio é calculada pela seguinte equação:

$$y = f\left(\sum_{i=1}^{N} x_i w_i \right) \tag{23}$$

Além disto, geralmente é inserida no modelo uma unidade especial com uma entrada constante $x_0 = 1$ e cujo peso w_0 é chamado de *bias*, com o intuito de aumentar ou diminuir (se w_0 é positivo ou negativo) o valor de entrada para a função de ativação (HAYKIN, 1999). Assim, teremos o valor limiar $T = -w_0$. Isto pode fazer com que o argumento da função de ativação caia dentro ou fora de uma certa faixa de valores.

Desta forma, a Equação (23) é rescrita como:

$$y = f\left(w_0 + \sum_{i=1}^{N} x_i w_i \right) \tag{24}$$

Os valores dos pesos w_i podem ser ajustados de forma que a rede tenha o comportamento desejado. Esse ajuste de parâmetros é conhecido como o *treinamento* da rede. Através da fase de treinamento, um conjunto de determinadas entradas é apresentado à rede, que extrai informações relacionadas ao ajuste dos pesos das sinapses, necessárias para generalizar o problema e gerar respostas coerentes para uma nova entrada futura.

5.5 Funções de Ativação

A função de ativação $f(z)$ define as propriedades computacionais de um neurônio. Uma função, para ser utilizada como função de ativação, deve apresentar comportamento monotônico sobre uma faixa determinada de valores de z e assumir um valor constante fora desta faixa de valores.

Diversas funções podem ser utilizadas como funções de ativação. HAYKIN (1999) e KÓVACS (2002) apresentam uma variedade de funções que podem ser utilizadas para a ativação do neurônio.

As funções de ativação mais utilizadas são:

Função Linear – A função linear é o exemplo mais simples de função de ativação (Figura 30). Esta função é utilizada no modelo Adaline, que é um modelo linear do funcionamento do neurônio biológico (WIDROW & HOFF, 1960).

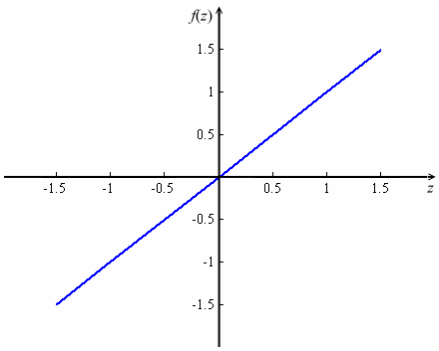

Figura 30. Gráfico da função linear.

Matematicamente, definimos a função linear por:

$$y = f(z) = z = \sum_i w_i x_i \tag{25}$$

No caso de uma função de ativação linear temos apenas o comportamento monotônico, desprezando o fato da função nunca assumir um valor constante fora de uma certa faixa de valores.

Função Lógica – A função lógica, também chamada de "threshold" é uma função do tipo degrau (ou sinal) (Figura 31). Esta função é utilizada no modelo perceptron (ROSENBLATT, 1958), e permite representar funcionalidades de natureza lógica realizadas pelos neurônios biológicos, como, por exemplo, a tomada de decisões (sim ou não).

Definimos a função lógica por:

$$y = f(z) = \begin{cases} 0, & z < \beta \\ 1, & z \geq \beta \end{cases} \tag{26}$$

onde o parâmetro β determina o ponto de transição (*threshold*) do neurônio.

A função de ativação lógica não é contínua em $z = \beta$, o que dificulta a utilização de algoritmos eficientes de treinamento.

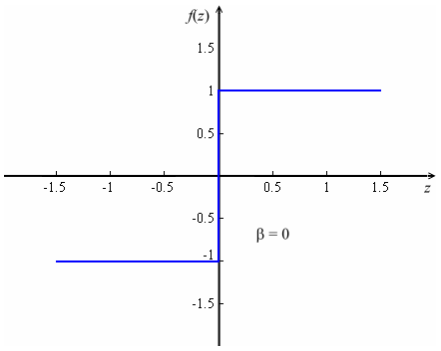

Figura 31. Gráfico da função lógica.

Função Sigmóide – A função sigmóide é uma versão contínua da função de ativação lógica. Esta função é uma das mais utilizadas em aplicações de redes neurais artificiais.

Uma função, para ser do tipo sigmóide precisa ser limitada, ou seja, nunca exceder um limite superior e um limite inferior, independente do valor de entrada. A função sigmóide deve ser monótona crescente, isto é, quando o valor de entrada cresce o valor da função cresce e vice-versa. Além disso, a função sigmóide é contínua, suave e derivável em todos os pontos.

Várias funções podem ser classificadas como funções sigmóide, como por exemplo, a função logística e a função tangente hiperbólica.

A função logística, muitas vezes chamada apenas de função sigmóide (Figura 32), é definida pela equação:

$$y = f(z) = \frac{1}{1+e^{-\alpha z}} \tag{27}$$

onde o parâmetro α modifica a derivada da função sigmóide nas vizinhanças do ponto $z = 0$ e serve para ajustar a "velocidade" da transição.

No limite tem-se:

$$\lim_{\alpha \to 0} y = \frac{1}{2}; \qquad \lim_{\alpha \to \infty} y = \begin{cases} 0, & z < 0 \\ 1, & z \geq 1 \end{cases} \tag{28}$$

Esta função, utilizada como função de ativação, limita a saída do neurônio no intervalo (0, 1), de forma que o valor das unidades de saída podem ser interpretadas como graus de pertinência em problemas de reconhecimento de padrões.

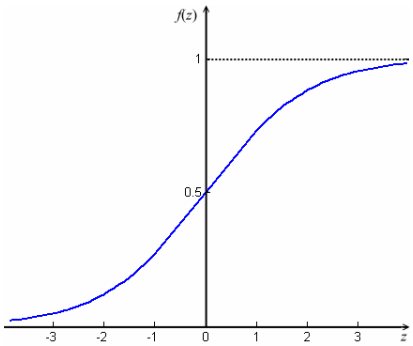

Figura 32. Gráfico da função sigmóide (logística).

Outra vantagem da função de ativação sigmóide é que a sua derivada é facilmente calculada:

$$\frac{dy}{dz} = f'(z) = \frac{\alpha\, e^{-\alpha z}}{\left(1+e^{-\alpha z}\right)^2} \tag{29}$$

Rearrumando a expressão temos:

$$\frac{dy}{dz} = f'(z) = \frac{\alpha}{\left(1+e^{-\alpha z}\right)}\frac{e^{-\alpha z}}{\left(1+e^{-\alpha z}\right)} = \frac{\alpha}{\left(1+e^{-\alpha z}\right)}\left(1-\frac{1}{\left(1+e^{-\alpha z}\right)}\right) \tag{30}$$

Assim:

$$\frac{dy}{dz} = f'(z) = \alpha\, y(1-y) \tag{31}$$

Em alguns casos, um melhor resultado no treinamento da rede é obtido com valores de saída limitados no intervalo (-1, 1). Para isso, utiliza-se a função de ativação tangente hiperbólica (Figura 33), definida pela equação:

$$y = f(z) = \frac{1-e^{-z}}{1+e^{-z}} \tag{32}$$

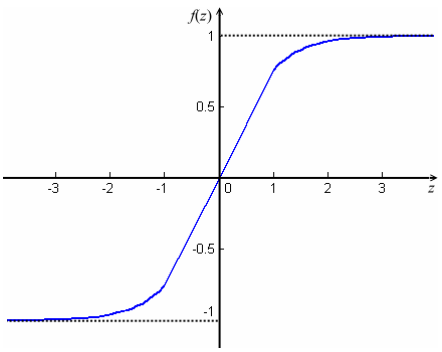

Figura 33. Gráfico da função tangente hiperbólica.

5.6 Tipos de Redes Neurais

Em geral, visualizamos os neurônios como arrumados em camadas. Tipicamente, neurônios na mesma camada comportam-se da mesma maneira. Os principais fatores na determinação do comportamento de um neurônio são sua função de ativação e o padrão das conexões através das quais ele envia e recebe sinais. Em cada camada, os neurônios usualmente possuem a mesma função de ativação e o mesmo padrão de conexões a outros neurônios.

A arrumação de neurônios em camadas (topologia) e o padrão das conexões entre as camadas é chamada de *arquitetura* da rede.

Podemos classificar uma rede neural de acordo com sua *arquitetura* e também de acordo com a *regra de aprendizado* utilizada.

5.6.1 Arquiteturas de Redes Neurais

A arquitetura de uma rede neural é determinada pela forma em que são definidas as conexões entre os neurônios. Basicamente, uma rede é dividida em camadas de neurônios. Dependendo do número de camadas que compõem a topologia da rede, as redes neurais podem ser classificadas como redes de camada simples (single layer) ou redes multicamadas (multilayer). Na determinação do número de camadas, as unidades de entrada não são contadas como uma camada, porque elas não realizam nenhum cálculo. Equivalentemente, o número de camadas na rede pode ser definido como o número de camadas de conexões entre os neurônios (FAUSETT, 1994). Assim, uma rede de camada simples é aquela que possui apenas uma camada de conexões entre a entrada e a saída da rede (Figura 34).

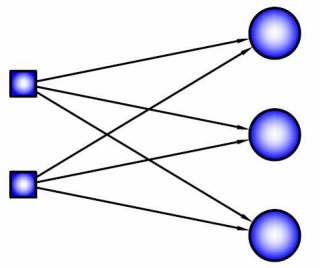

Figura 34. Rede Neural de camada simples.

Uma rede multicamadas é aquela que possui mais de uma camada de conexões entre a entrada e a saída da rede, tendo, portanto, pelo menos uma camada intermediária

de neurônios. A camada intermediária de neurônios é também conhecida como camada escondida (*hidden*).

Uma rede multicamadas, embora seja mais complexa, apresenta melhores resultados e, portanto, possui mais aplicações. Uma rede neural multicamadas pode ser vista como uma rede composta por várias redes de camadas simples, na qual os neurônios de camadas adjacentes são conectados. A Figura 35 ilustra uma rede neural com uma camada intermediária de neurônios.

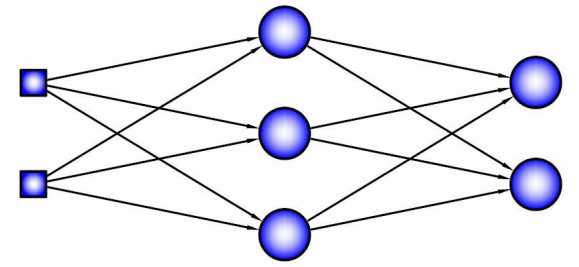

Figura 35. Rede Neural multicamadas.

A topologia das redes neurais multicamadas possibilitou a resolução de problemas cujas classes não são linearmente separáveis.

A complexidade do modelo que pretende-se aproximar pela rede neural irá definir a quantidade de pesos (parâmetros) e de camadas da arquitetura da rede. O número de neurônios não deve ser muito pequeno se o problema a ser resolvido apresentar grande complexidade e nem grande demais, a ponto de se prejudicar a capacidade de generalização da rede, gerando o problema conhecido como *overfitting* (ajuste excessivo).

Estruturalmente, as redes neurais podem ser divididas em: redes *feedforward* e redes recorrentes.

Redes Feedforward – Nas redes *feedforward*, também chamadas de acíclicas, a saída de um neurônio não pode ser utilizada como entrada de outro neurônio em nenhuma camada anterior, ou seja, as ligações sinápticas só permitem a passagem de estímulos aos neurônios das camadas seguintes. Como exemplos clássicos de redes *feedforward* temos o *Perceptron* e o *Adaline*. As redes neurais apresentadas nas Figuras 34 e 35 são redes *feedforward*.

Redes Recorrentes – Nas redes recorrentes, também chamadas de cíclicas, a saída de um neurônio pode ser utilizada como entrada para neurônios da mesma camada, incluindo o próprio neurônio formando um loop (*self-feedback*), ou para neurônios de

camadas anteriores (*feedback*). Exemplos de redes recorrentes foram apresentados em (ANDERSON, 1977; KOHONEN, 1977; HOPFIELD, 1982). Um exemplo de rede recorrente (competitiva) pode ser visto na Figura 36.

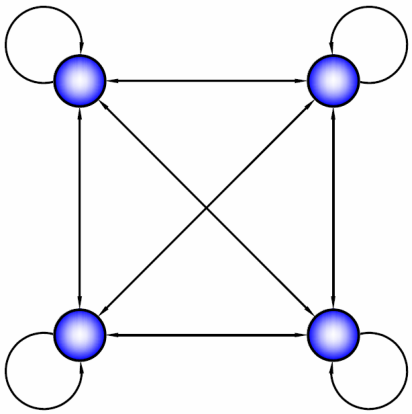

Figura 36. Rede recorrente.

5.6.2 Aprendizado de Redes Neurais

Uma rede neural precisa ser configurada de maneira que a aplicação de um conjunto de entradas produza o conjunto de saídas desejado. Para isso, a rede passa por um processo de aprendizado através de apresentações repetidas de um conjunto de exemplos (pares entrada-saída) de treinamento à rede. Cada apresentação de todo um conjunto de treinamento durante o processo de aprendizagem é chamada de *época*. O processo de aprendizagem é repetido época após época, até que um determinado critério de parada seja atingido. Os estilos de treinamento de pesos e *bias* mais tradicionais são: treinamento local ou incremental, e treinamento em lote ou *batch*.

No treinamento local ou incremental os pesos e o *bias* da rede são atualizados imediatamente após a apresentação de cada entrada à rede. Entretanto, no treinamento em lote ou *batch* a atualização de pesos e *bias* da rede é feita só depois que todas as entradas forem apresentadas à rede (*época*). A eficiência dos dois métodos depende do problema a ser tratado (HAYKIN, 1999).

Podemos categorizar o processo de aprendizado como: supervisionado, por reforço ou não-supervisionado.

Aprendizado Supervisionado – O aprendizado supervisionado é o processo de aprendizado no qual a rede é treinada através do fornecimento dos valores de entrada e seus respectivos valores de saída desejados (*"training pair"*). Esse conjunto de valores de entrada e saída é denominado *conjunto de treinamento*. O ajuste dos pesos sinápticos

da rede é realizado através do processo de minimização do erro calculado a partir da comparação entre a saída desejada e a saída obtida pela rede. Os pesos da rede são ajustados sob influência combinada do vetor de entrada e o sinal de erro. Com o aprendizado supervisionado, o conhecimento disponível ao agente externo é indiretamente transferido à rede neural. Os principais algoritmos de aprendizado supervisionado são a Regra Delta e o algoritmo de Retropropagação.

Aprendizado por Reforço – O aprendizado por reforço é um caso particular do aprendizado supervisionado (SUTTON & BARTO, 1998). Neste tipo de aprendizado, o comportamento da rede é avaliado simplesmente com base em algum critério numérico, fornecido em instantes espaçados de tempo. Este procedimento procura maximizar um índice escalar de desempenho chamado sinal de reforço, onde ocorre a avaliação indireta do comportamento, não fornecendo indicações se o melhoramento é possível, ou de como o sistema deverá proceder para melhor atender os objetivos (GUARIZE, 2004).

Aprendizado Não-Supervisionado – O aprendizado não-supervisionado é o processo de aprendizado no qual apenas os padrões de entrada são fornecidos, ou seja, não requer o valor desejado de saída da rede. O sistema extrai as características do conjunto de padrões, agrupando-os em classes inerentes aos dados. Uma vez que a rede conclui um processo de auto-organização (*Self-Organization*) baseado nos dados de entrada, ela explora a representação interna gerada para discriminar características presentes na entrada. Os principais algoritmos de aprendizado não-supervisionado são aprendizado Hebbiano e aprendizado por competição.

5.6.3 Algoritmos de Treinamento

No processo de treinamento supervisionado em redes neurais multicamadas podem ser identificados processos de minimização de uma função não-linear. O método adotado para minimização de erro quadrático no processo de treinamento é escolhido em função dos problemas abordados e dos padrões de treinamento escolhidos.

A título de exemplificação, pode-se destacar alguns métodos adotados: algoritmo de retropropagação ou gradiente descendente, gradiente conjugado, Newton (HAYKIN, 1999), Levenberg-Marquardt (HAGAN & MENHAJ, 1994), método de pontos interiores (TRAFALIS & COUELLAN, 1994; LEMMON & SZYMANSKI, 1994),

algoritmos genéticos (IYODA *et al.*, 1999; VELAZCO & LYRA, 2002) e método de enxame de partículas (SETTLES & RYLANDER, 2002).

A seguir serão descritos os algoritmos de retropropagação (gradiente descendente), gradiente conjugado e Levenberg-Marquardt.

O Algoritmo de Retropropagação

O Algoritmo de Retropropagação (*Backpropagation*), também conhecido como o método do Gradiente Descendente (*Steepest Descent*), é um algoritmo utilizado no treinamento de redes neurais multicamadas com uma ou mais camadas escondidas.

Basicamente, o algoritmo de retropropagação consiste em dois passos de computação: o processamento direto (*forward*) e o processamento reverso (*backward*). No processamento direto, uma entrada é aplicada à rede neural, seu efeito é propagado pela rede através das conexões, cujos pesos permanecem inalterados, até que uma resposta seja produzida pela camada de saída. No processamento reverso, a resposta obtida da rede é comparada com a saída desejada e, caso não esteja correta, o erro é calculado e é retropropagado da camada de saída à entrada, ajustando assim os pesos das conexões nas unidades das camadas internas. O processo é repetido até que atinja um critério de parada pré-estabelecido.

A maioria das aplicações de redes neurais artificiais utiliza redes de duas camadas. Por exemplo, CYBENKO (1989) demonstrou rigorosamente que apenas uma única camada intermediária de neurônios seria suficiente para aproximar funções contínuas através de redes neurais.

Para iniciar o processo de aprendizado pelo algoritmo de retropropagação, é necessário estabelecer:

a) Um conjunto de padrões de entrada e suas respectivas saídas esperadas;

b) Uma taxa de aprendizado η;

c) Um critério de parada, com o qual se dará o treinamento como concluído;

d) Uma função de ativação do neurônio.

A seguir, será descrito como funciona o algoritmo de retropropagação aplicado a uma rede de duas camadas, ou seja, uma rede com apenas uma camada intermediária de neurônios.

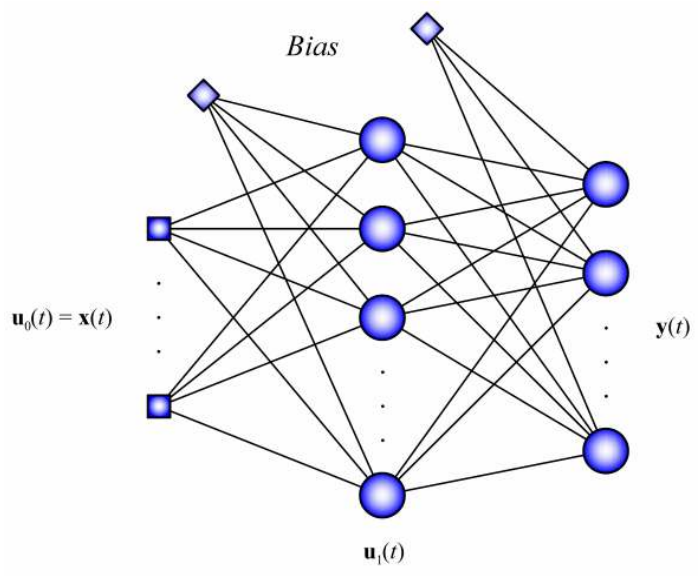

Figura 37. Rede neural com uma camada intermediária de neurônios.

Considere a rede representada na Figura 37 contendo *p* unidades de entrada, *n* unidades na camada intermediária e *m* unidades na camada de saída. Suponha que o conhecimento sobre a solução do problema possa ser representado pelo conjunto de treinamento $T = \{(\mathbf{x}(t), \mathbf{v}(t)), t = 1...N\}$, onde, para cada exemplo *t*, **x**(t) representa o vetor de entradas e **v**(t) o vetor de saídas desejadas. A rede neural deve ser treinada de modo que seja minimizado o erro entre as saídas corretas **v**(t) e as saídas calculadas pela rede **y**(t).

No primeiro passo do algoritmo de retropropagação, trabalhamos apenas com as informações do vetor de entradas $\mathbf{u}_0(t) = \mathbf{x}(t) = \{x_1(t),...,x_p(t)\}$, fazendo a propagação dessas informações pela rede através das conexões até a camada de saída. Os valores das unidades da camada intermediária são calculados de acordo com o vetor de entradas $\mathbf{u}_0(t) = \mathbf{x}(t)$ e com os pesos das suas respectivas conexões com as unidades de entrada. Seja w_{ji}^k o parâmetro que representa o peso da conexão entre um neurônio *i* na camada *k* e um neurônio *j* na camada $k+1$, e seja $\boldsymbol{\beta}_{k\text{-}1}(t) = \{\beta_1^{k-1}(t),...,\beta_n^{k-1}(t)\}$ o vetor que define os valores das conexões entre o vetor *bias* da camada $k-1$ e a saída da camada *k*. A partir da entrada $\mathbf{u}_0(t) = \mathbf{x}(t)$, a saída da camada intermediária $\mathbf{u}_1(t) = \{u_1^1(t),...,u_n^1(t)\}$ é calculada como:

$$u_i^1(t) = f\left(\beta_i^0(t) + \sum_{j=0}^{p} w_{ji}^0 u_j^0(t)\right) \tag{33}$$

Da mesma maneira, as unidades da camada de saída $\mathbf{y}(t) = \{y_1(t),...,y_m(t)\}$ são calculadas a partir da camada intermediária $\mathbf{u}_1(t)$ como:

$$y_i(t) = f\left(\beta_i^1(t) + \sum_{j=0}^{n} w_{ji}^1 u_j^1(t)\right) \tag{34}$$

A partir das equações (33) e (34), podemos ver que, para uma estrutura de rede neural com qualquer número de camadas, a saída de uma camada é calculada em função da saída da camada anterior. Assim sendo, temos que a fórmula geral de como os valores de entrada são propagados na rede através das conexões, até que uma resposta seja produzida pela camada de saída é dada por:

$$u_i^k(t) = f\left(\beta_i^{k-1}(t) + \sum_{j=0}^{c} w_{ji}^{k-1} u_j^{k-1}(t)\right) \tag{35}$$

onde *c* é o número de unidades na camada *k*.

No segundo passo do algoritmo de retropropagação, trabalhamos com a informação do vetor de saídas desejadas $\mathbf{v}(t) = \{v_1(t),\ldots,v_m(t)\}$. Assim, o sinal de erro para o neurônio *j*, definido como a diferença entre a resposta desejada e a resposta observada, é dado pela equação:

$$e_j(t) = y_j(t) - v_j(t) \tag{36}$$

O objetivo do procedimento de aprendizado por correção de erro é minimizar alguma *função de custo* baseada no sinal do erro $e_j(t)$, de modo que a resposta observada de cada neurônio da rede se aproxime da resposta desejada para aquele neurônio, em algum sentido estatístico. De fato, uma vez definida uma função de custo, o problema de aprendizado torna-se um problema de otimização.

A fórmula do erro instantâneo é dada pela soma dos erros quadráticos da rede:

$$E(t) = \frac{1}{2}\sum_{j=1}^{m}\left(e_j(t)\right)^2 \tag{37}$$

Se *N* é o número total de exemplos no conjunto de treinamento, a função do *erro médio quadrático* é obtida somando-se *E*(*t*) para todo valor de *t*, e então normalizando-se o resultado com relação a *N*, como segue:

$$E = \frac{1}{N}\sum_{t=1}^{N} E(t) \tag{38}$$

O erro instantâneo $E(t)$ e o erro médio quadrático E são função de todos os parâmetros livres (pesos e *bias*) da rede. Para um dado conjunto de treinamento, E representa a *função de custo* como uma medida da performance do aprendizado, ou seja, a rede neural aprende através da minimização de E em relação aos pesos sinápticos da rede.

Para minimizar o erro médio quadrático, necessitamos primeiramente determinar o gradiente instantâneo, que é a equação da derivada do erro instantâneo $E(t)$ em relação ao peso sináptico $w_{ji}(t)$ do neurônio j da camada de saída.. Aplicando a regra da cadeia, podemos expressar este gradiente como:

$$\frac{\partial E(t)}{\partial w_{ji}(t)} = -e_j(t) f'\left(z_j^1(t)\right) y_i(t) = \delta_j(t) y_i(t) \tag{39}$$

sendo:

$$z_j^1(t) = \beta_i^1(t) + \sum_{j=0}^{n} w_{ji}^1 u_j^1(t) \tag{40}$$

e

$$\delta_j(t) = -e_j(t) f'\left(z_j^1(t)\right) = \left(y_j(t) - v_j(t)\right) f'\left(z_j^1(t)\right) \tag{41}$$

onde $\delta_j(t)$ é chamado de gradiente local.

Observe que a Equação (39) corresponde a um componente do vetor gradiente do erro, cujos elementos representam a derivada parcial de $E(t)$ em relação a todos os pesos da rede neural, arranjados em uma ordem fixa, mas arbitrária.

Pelo algoritmo do gradiente, a cada iteração do processo de treinamento, a adaptação dos pesos das conexões é proporcional à derivada da função erro em relação a $w_{ji}(t)$ de acordo com a chamada *regra delta*:

$$w_{ji}(t+1) = w_{ji}(t) - \eta \delta_j(t) y_i(t) \tag{42}$$

onde o parâmetro η é chamado de taxa de aprendizagem e determina quanto o valor do parâmetro $w_{ji}(t)$ deve "caminhar" na direção contrária à derivada (máxima variação do erro).

No entanto, quando o neurônio pertence à camada intermediária (escondida), não há nenhuma saída desejada pré-especificada para o neurônio. Assim, o sinal do erro deve ser calculado em termos dos sinais de erro de todos os neurônios aos quais o neurônio escondido está diretamente conectado. Temos, então, que a expressão para o gradiente local para um neurônio *j* da camada escondida é dada por:

$$\delta_j(t) = f'\left(z_j^0(t)\right)\sum_{k=1}^{n}\delta_k(t)w_{kj}(t) \tag{43}$$

onde

$$z_j^0(t) = \beta_i^0(t) + \sum_{j=0}^{p} w_{ji}^0 u_j^0(t) \tag{44}$$

O somatório na Equação (43) depende de dois conjuntos de termos: os $\delta_i(t)$ exigem o conhecimento dos sinais de erro $e_k(t)$, para todos os neurônios localizados na camada imediatamente posterior à camada onde se encontra o neurônio *j*, e que estão diretamente conectados ao neurônio *j*; e os $w_{kj}(t)$, que consistem nos pesos sinápticos associados a estas conexões (IYODA, 2000).

Utilizando esses novos valores de pesos, a idéia do método é repetir o processo iterativamente, até que o erro nas saídas da rede seja menor que um valor máximo aceitável.

Este método não apresenta garantia de convergência (KÓVACS, 2002). Pelo fato de adotar a direção de busca do mínimo como sendo a mesma do vetor gradiente, o custo computacional deste método também se torna elevado. Existem várias variações deste método na literatura que tentam melhorar a sua convergência bem como sua eficiência computacional (KÓVACS, 2002; SHEWCHUK, 1994; ZHU, 2003). Em linhas gerais, este método não é um procedimento robusto de solução (MATOS, 2005).

O Método do Gradiente Conjugado

O algoritmo de retropropagação (*backpropagation*) usando o método do gradiente descendente freqüentemente converge muito lentamente ou mesmo não converge. Em problemas de larga escala (com muitos pesos para serem ajustados), seu sucesso depende do valor da taxa de aprendizado, fornecido pelo usuário. Não existe uma maneira automática de selecionar esse parâmetro e, se um valor incorreto for especificado, a convergência pode ser extremamente lenta, ou o método pode não

convergir. Embora o *backpropagation* com gradiente descendente ainda seja usado em muitos programas de redes neurais, ele não é mais considerado ser o melhor algoritmo e nem o mais rápido.

Como uma melhor opção temos o método do gradiente conjugado, que pertence à classe dos métodos de otimização de segunda ordem, conhecidos coletivamente como método de direção conjugada (HAYKIN, 1999).

Vários algoritmos de gradiente conjugado foram propostos como algoritmos de aprendizado em redes neurais (BATTITI, 1992; MOLLER, 1993). JOHANSSON *et al.* (1990) descreveu a teoria dos métodos gerais de gradiente conjugado e como aplicar os métodos em redes neurais *feedforward*. Comparado com o gradiente descendente, o algoritmo do gradiente conjugado toma um caminho mais "direto" até um conjunto ótimo de valores para os pesos da rede. Usualmente, o gradiente conjugado é significantemente mais rápido e mais robusto que o gradiente descendente (TOWSEY, 1995). O método do gradiente conjugado também não requere que o usuário especifique o valor da taxa de aprendizado.

O algoritmo do gradiente conjugado tradicional usa o gradiente para computar uma direção de busca. Então, utiliza um algoritmo de busca em linha (*line search*) como, por exemplo, o *Método de Brent* (BRENT, 1973), para encontrar o tamanho ótimo do passo ao longo de uma linha na direção de busca. A busca em linha evita a necessidade de se computar a matriz Hessiana da derivada segunda, mas necessita da computação do erro em vários pontos ao longo da linha. O algoritmo do gradiente conjugado com busca em linha tem sido utilizado com sucesso em muitos problemas de redes neurais, e é considerado um dos melhores métodos já inventados.

Um novo algoritmo de gradiente conjugado, chamado de método do gradiente conjugado escalonado (*Scaled Conjugate Gradient*), foi desenvolvido por MOLLER (1993) para evitar a busca em linha em cada iteração de aprendizado (GUPTA *et al.*, 2003). Se $\mathbf{w}$ é o vetor de pesos da rede, a forma da equação básica de atualização dos pesos do algoritmo do gradiente conjugado é dada por:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta(k)\Delta\mathbf{w}(k) \tag{45}$$

onde $\eta(k)$ é o parâmetro de aprendizado que varia no tempo que pode ser atualizado usando o seguinte método de busca em linha:

$$\eta(k) = \arg\min\{E(\mathbf{w}(k) + \eta\Delta\mathbf{w}(k)) : \eta \geq 0\} \tag{46}$$

A condição conjugada para o incremento do vetor de pesos $\Delta\mathbf{w}$ é dada por (SMAGT, 1994):

$$\left(\Delta\mathbf{w}(k)\right)^T \boldsymbol{H}(k)\Delta\mathbf{w}(k+1) = 0 \tag{47}$$

onde a matriz Hessiana $\boldsymbol{H}(k)$ é calculada no ponto $\mathbf{w}(k)$. A atualização para $\Delta\mathbf{w}(k)$, nesse caso, é escolhida como (SMAGT, 1994):

$$\Delta\mathbf{w}(k) = -\boldsymbol{g}(k) + \alpha(k-1)\Delta\mathbf{w}(k-1) \tag{48}$$

onde $\boldsymbol{g}(k)$ é o vetor gradiente. Na primeira iteração do algoritmo, ou seja, quando $k = 1$ temos que $\Delta\mathbf{w}(1) = -\boldsymbol{g}(1)$.

Para satisfazer a condição conjugada entre os vetores $\Delta\mathbf{w}(k)$ e $\Delta\mathbf{w}(k+1)$, as expressões para a atualização do parâmetro $\alpha(k)$ são dadas por:

(i) Formulação de Fletcher-Reeves:

$$\alpha(k) = \frac{\left(\boldsymbol{g}(k+1)\right)^T \boldsymbol{g}(k+1)}{\left(\boldsymbol{g}(k)\right)^T \boldsymbol{g}(k)} \tag{49}$$

(ii) Formulação de Polak-Ribière:

$$\alpha(k) = \frac{\left(\boldsymbol{g}(k+1)\right)^T \left[\boldsymbol{g}(k+1) - \boldsymbol{g}(k)\right]}{\left(\boldsymbol{g}(k)\right)^T \boldsymbol{g}(k)} \tag{50}$$

(iii) Formulação de Hestenes-Stiefel:

$$\alpha(k) = \frac{\left[\boldsymbol{g}(k) - \boldsymbol{g}(k-1)\right]^T \boldsymbol{g}(k)}{\left(\Delta\mathbf{w}(k-1)\right)^T \left[\boldsymbol{g}(k) - \boldsymbol{g}(k-1)\right]} \tag{51}$$

Entretanto, a melhor fórmula para a atualização de α é altamente dependente do problema estudado. Muitos estudos indicam que os métodos de Polak-Ribière e de Hestenes-Stiefel produzem uma melhor performance. O algoritmo de aprendizado por gradiente conjugado é um tipo especial de algoritmo de aprendizado *backpropagation*, onde a informação das derivadas parciais de segunda-ordem são usadas para atualizar a taxa de aprendizado. De fato, o algoritmo do gradiente conjugado precedente utiliza informações sobre a direção de busca para $\Delta\mathbf{w}$ da iteração anterior a fim de acelerar a convergência. Cada direção de busca é conjugada se a função objetivo é quadrática.

O Algoritmo de Levenberg-Marquardt

O algoritmo de Levenberg-Marquardt foi desenvolvido para resolver iterativamente problemas de minimização de funções não-lineares pelo método de mínimos quadrados (LEVENBERG, 1944; MARQUARDT, 1963). Este método é bastante eficiente quando aplicado no treinamento de redes *feedforward* que não possuem mais do que algumas centenas de pesos sinápticos a serem ajustados (HAGAN & MENHAJ, 1994). É o algoritmo de otimização mais utilizado atualmente, superando o método do gradiente descendente e os outros métodos de gradiente conjugado em uma grande variedade de problemas.

Para acelerar o treinamento, o algoritmo de Levenberg-Marquardt utiliza a informação das derivadas de segunda ordem do erro quadrático em relação aos pesos, assim como o método do gradiente conjugado.

De maneira geral, o algoritmo de Levenberg-Marquardt é uma combinação do algoritmo de retropropagação com gradiente descendente com o método iterativo de Gauss-Newton, que faz uso da matriz Hessiana **H**. No método de Levenberg-Marquardt faz-se uma aproximação para essa matriz, dada pela equação (52), determinada em função da matriz Jacobiana, que contém as primeiras derivadas dos erros em função dos pesos sinápticos (Equação 53).

$$\mathbf{H} = \frac{\partial^2 E(\mathbf{w})}{\partial \mathbf{w}^2} \tag{52}$$

$$\mathbf{J} = \frac{\partial e(\mathbf{w})}{\partial \mathbf{w}} \tag{53}$$

onde E é o erro médio quadrático e $e(\mathbf{w})$ é definido conforme a expressão:

$$e(\mathbf{w}) = \sum_{i=1}^{n} \left(y_i - v_i \right) \tag{54}$$

onde y_i é a saída fornecida pela rede e v_i é o valor exato correspondente à saída da rede.

Determinar a matriz Jacobiana é muito mais simples que determinar a matriz Hessiana. Portanto, como, para uma rede neural, a performance de treinamento é expressa em função da soma dos erros quadráticos, a matriz Hessiana pode ser dada pela equação:

$$\mathbf{H} = \mathbf{J}^{\mathrm{T}}(\mathbf{w}).\mathbf{J}(\mathbf{w}) \tag{55}$$

O método de Newton atualiza os pesos segundo:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mathbf{H}^{-1}.g_k \tag{56}$$

onde g_k pode ser escrito conforme:

$$g_k = 2\mathbf{J}^{\mathrm{T}}(\mathbf{w}).e(\mathbf{w}) \tag{57}$$

O algoritmo de Levenberg-Marquardt procede a atualização dos pesos baseado na mesma expressão do método de Newton (Equação 56), realizando as modificações para a determinação da matriz Hessiana, como segue:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \left(\mathbf{H} + \mu_k \mathbf{I}\right)^{-1}.\mathbf{J}^{\mathrm{T}}(\mathbf{w}).e(\mathbf{w}) \tag{58}$$

onde **I** é a matriz identidade e μ_k é o fator de ajuste do método de Levenberg-Marquardt.

Podemos ver que a regra de atualização leva em consideração tanto a inclinação da superfície do erro (método do gradiente descendente) quanto a curvatura desta superfície (método de Gauss-Newton). O fator de ajuste μ_k indica qual dos dois métodos será predominante: para fatores de ajuste grandes, o método do gradiente descendente predomina e a atualização dos pesos ocorre fortemente na direção de inclinação da superfície do erro; caso contrário, o método de Gauss- Newton predomina e a atualização ocorre mais no sentido da curvatura da função. O algoritmo controla o valor do fator de ajuste da seguinte maneira: começa-se com um valor arbitrário. Calcula-se o erro na situação atual e aplica-se a regra de atualização de pesos. Calcula-se, então, o novo erro. Caso o erro tenha aumentado, deve-se desfazer a atualização e aumentar o fator de ajuste (geralmente multiplicando-o por dez). Em seguida, deve-se recomeçar a iteração novamente. Caso o erro tenha diminuído, a iteração é aceita e diminui-se o fator de ajuste (geralmente dividindo-o por dez) (RANGANATHAN, 2004; WINANDY, *et al.*, 2007).

Desta maneira, os valores irão "caminhar" mais na direção do gradiente quanto mais distante estiverem do ponto mínimo. Ao chegar às proximidades dele, o algoritmo de Gauss-Newton será predominante, o que faz com que o algoritmo de Levenberg-Marquardt funcione utilizando o que cada um dos algoritmos anteriores tem de melhor.

No entanto, há problemas no algoritmo de Levenberg-Marquardt, como, por exemplo, a dificuldade de se calcular a matriz Hessiana da função erro e a matriz

inversa presente na regra de atualização. Além disso, outro ponto de dificuldade do algoritmo é que, para valores muito elevados do fator de

ajuste, o cálculo da matriz Hessiana é praticamente desprezado. Esse último ponto levou Marquardt a propor uma melhoria no algoritmo original que faz com que a parcela relativa ao gradiente descendente também passe a incorporar informação da curvatura da superfície do erro, fazendo com que a atualização de pesos seja considerável mesmo quando o gradiente é bem pequeno. A regra final de atualização do algoritmo de Levenberg-Marquardt é então dada pela equação:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \left(\mathbf{H} + \mu_k \, diag\left[\mathbf{H}\right]\right)^{-1} . \mathbf{J}^{\mathrm{T}}(\mathbf{w}) . e(\mathbf{w}) \tag{59}$$

Esse método apresenta convergência em menos iterações, mas requer mais cálculos por iteração devido ao cálculo de matrizes inversas. Apesar do grande esforço computacional, ele segue sendo o algoritmo de treinamento mais rápido para redes neurais, quando se trabalha com um número moderado de parâmetros na rede. Se esse número é elevado, a utilização desse algoritmo é pouco prática.

5.6.4 Generalização da Rede Neural

Ao tentar minimizar o erro no conjunto de treinamento se está, de certa forma, fazendo a rede "decorar" o comportamento neste conjunto ao invés de "aprender" o comportamento do problema. Este processo de minimização pode pôr em risco a capacidade de generalização da rede neural conduzindo a um ajuste excessivo junto ao conjunto de treinamento, que certamente representa um conjunto finito de informação e pode conter amostras sujeitas a ruído, que passariam a ser incorporados à solução (HAYKIN, 1999; WASSERMAN, 1989; MITCHELL, 1997). Tal ajuste excessivo é conhecido como *overfitting*.

O processo de aprendizagem da rede, isto é, o treinamento da rede, pode ser considerado um problema de "ajuste de curva". Uma rede com boa capacidade de generalização é aquela que realiza um mapeamento não-linear de entrada-saída computado pela rede de forma correta (ou aproximadamente correta) para dados de teste não utilizados no treinamento. As Figuras 38 e 39 mostram esquematicamente uma rede treinada adequadamente e uma rede treinada em excesso (*overfitting*), respectivamente.

Em linhas gerais a generalização é influenciada por três fatores:

- O tamanho do conjunto de treinamento, e quão representativo do ambiente de interesse ele é;

- A arquitetura da rede neural;
- A complexidade física do problema em questão (evidentemente não há controle sobre este fator).

Na definição da topologia de uma rede neural, normalmente, o número de camadas e o número de nós em cada camada são definidos em função de uma inspeção prévia nos dados e na complexidade do problema. Uma vez definida a topologia inicial, a estrutura final mais adequada para a modelagem é geralmente obtida através de refinamentos sucessivos, que podem levar a um tempo de dimensionamento alto, já que este tem um grande componente empírico.

O objetivo desta etapa de ajuste é a obtenção de uma topologia de rede que modele com precisão os dados do conjunto de treinamento, mas que também resulte em uma aproximação com boa capacidade de generalização (GUARIZE, 2004).

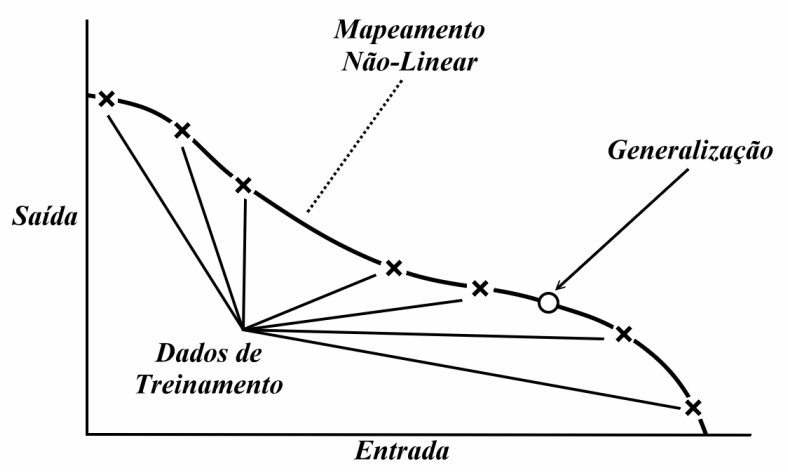

Figura 38. Rede treinada adequadamente (boa generalização).

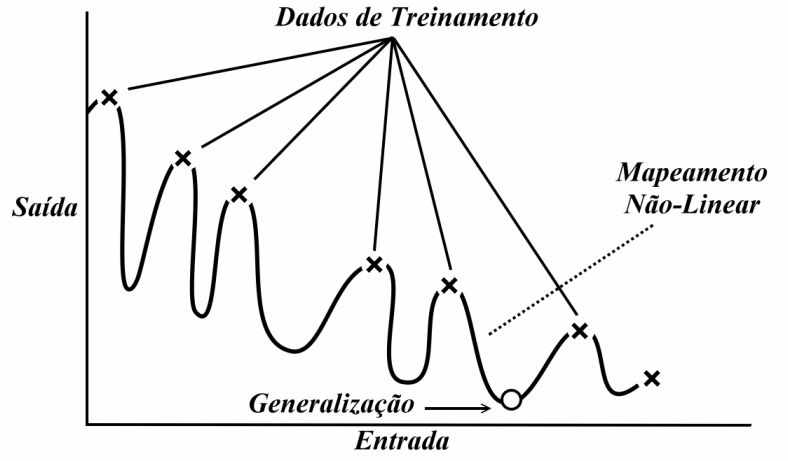

Figura 39. Rede treinada em excesso (*overfitting* – generalização pobre).

Há várias maneiras de se evitar o *overfitting*. Uma delas é denominada técnica de parada antecipada (ou *early-stopping*). A idéia é terminar o treinamento antes que a fase de ajuste fino tenha início e, por conseguinte, o *overfitting*. Para detectar este momento faz-se uso de um segundo conjunto de dados chamado conjunto de validação. O conjunto de validação, idealmente não deve diferir estatisticamente do conjunto de treinamento, mas também não pode ser muito similar. O conjunto de validação não será usado para treinar a rede, mas apenas para comparar o erro produzido pelo conjunto de treinamento com o erro produzido pelo conjunto de teste. Como fazem parte do mesmo problema, a tendência é que os erros de ambos os conjuntos diminuam ao longo do treinamento, porém, a partir de certo ponto, o erro do conjunto de treinamento continuará a diminuir (pois o algoritmo de aprendizado forçará esse comportamento) enquanto que o erro do conjunto de validação aumentará. Isso acontece porque o conjunto de validação não influencia o aprendizado da rede. Sendo assim, no momento em que a rede começar a se adequar melhor ao conjunto de treinamento do que ao conjunto de validação (um subconjunto qualquer dos casos do problema), isso significa que a rede começou a perder capacidade de generalização e, portanto, o treinamento deve ser interrompido (critério de parada). A Figura 40 ilustra esse comportamento.

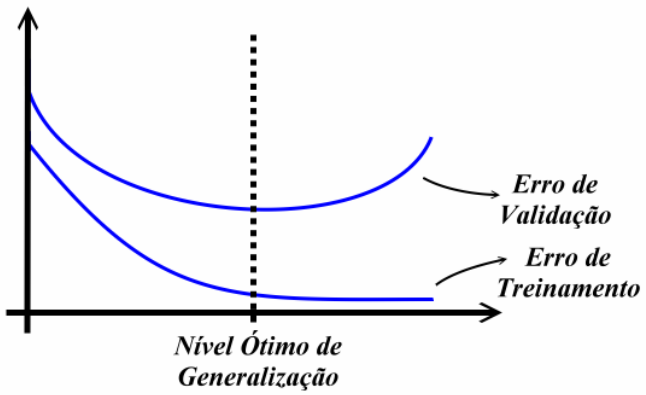

Figura 40. Comportamento do erro nos conjuntos de treinamento e de validação.

O conjunto de teste também não deve diferir dos outros dois conjuntos já mencionados anteriormente. Porém o conjunto de teste não exerce nenhuma influência no aprendizado da rede servindo mais para comparação de modelos diferentes ou de arquiteturas diferentes de redes que pretendem resolver o mesmo problema (ROCHA, 2007).

6 CUSTOMIZAÇÃO DO PSO APLICADO AO PROJETO DE *RISERS*

Neste capítulo serão apresentados estudos sobre a aplicação de uma estratégia de otimização promissora baseada em conceitos evolucionários ao projeto de *risers*: o método de Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO).

Apesar de ser relativamente recente, o método de PSO foi usado para a solução de uma variedade de problemas difíceis de otimização, apresentando em alguns casos uma taxa de convergência mais rápida do que outros algoritmos evolucionários (HU *et al.*, 2003, HASSAN *et al.*, 2005). Alguns exemplos de uso do PSO em aplicações diversas podem ser encontrados em (PENG *et al.*, 2000; ABIDO, 2002; OURIQUE *et al.*, 2002; KANNAN *et al.*, 2004; PARSOPOULOS & VRAHATIS, 2004; EBERHARDT & SHI, 2004; MEDEIROS, 2005). Na área de sistemas *offshore*, ALBRECHT (2005) e MONTEIRO (2008) empregaram o método de PSO na otimização de linhas de ancoragem.

Além de sua eficiência, uma outra razão para a popularidade crescente do método de PSO é que precisam ser ajustados menos parâmetros do que na maioria dos outros algoritmos evolucionários (EA), o que o torna particularmente fácil de implementar (HE *et al.*, 2004). Entretanto, como em todo EA, seu desempenho dependente dos valores selecionados para os parâmetros. Uma má escolha dos valores pode conduzir à convergência prematura a um ótimo local, ou causar um comportamento oscilatório com convergência lenta próximo à solução ótima. Algumas diretrizes básicas para a seleção de valores de parâmetros são apresentadas por TRELEA (2003), junto com uma análise teórica do algoritmo de PSO. Em todo caso, o comportamento do método é certamente dependente do problema, e análises paramétricas devem ser executadas para ajustá-lo para cada aplicação particular do algoritmo de PSO.

No que diz respeito a convergência, o comportamento do algoritmo de otimização é particularmente importante para a aplicação considerada neste capítulo - *risers* para a produção de petróleo *offshore*. Sabe-se que a avaliação do comportamento estrutural de configurações de *risers* exige análises estruturais com elementos finitos que empregam um "solver" dinâmico não-linear no domínio do tempo, que é extensivamente demorado. Logo, uma abordagem ideal de otimização deveria poder encontrar uma solução ótima com menos análises para o cálculo da *fitness* de cada configuração candidata, assim minimizando o esforço computacional.

Portanto, o objetivo deste capítulo é estudar algumas variações propostas para a formulação padrão do método de PSO, com o objetivo de realizar uma análise da convergência do algoritmo através de muitos experimentos, testando diferentes ajustes de parâmetros. O método foi implementado em uma ferramenta computacional destinada ao projeto de sistemas *offshore*, a ferramenta *ProgOtim*. Esta ferramenta incorpora outros métodos de otimização baseados em conceitos evolucionários apresentados em trabalhos anteriores (VIEIRA *et al.*, 2003; de LIMA *et al.*, 2005; ALBRECHT, 2005; VIEIRA, 2008; MONTEIRO, 2008; VIEIRA *et al.*, 2008a,b; PINA *et al.*, 2008; VIEIRA, 2009), incluindo algoritmos genéticos, micro algoritmos genéticos, PSO e estratégias evolucionárias, aplicadas ao projeto de diferentes sistemas *offshore* que incluem *risers* e linhas de ancoragem para sistemas de produção flutuantes.

Os resultados aqui apresentados foram publicados no periódico *Optimization and Engineering* (PINA *et al.*, 2010a).

6.1 Definição do Problema de Otimização do *Riser Lazy-Wave*

6.1.1 Variáveis de Projeto

A Figura 41 apresenta um modelo esquemático que mostra os parâmetros ou as variáveis de projeto que são consideradas para o processo de otimização do sistema de *risers Lazy-Wave*, para ser instalado em um cenário em que a lâmina d'água seja indicada por H.

Os parâmetros geométricos do *riser* são (**L1**), comprimento do segmento superior do *riser*; (**L2**), comprimento do segmento com flutuadores distribuídos; (**L3**), comprimento do segmento inferior do *riser*; (**α**) o "ângulo no topo", ou o ângulo do *riser* com relação ao eixo vertical na conexão com a plataforma, medido na configuração neutra de equilíbrio; (**z**) a profundidade da conexão, e (**P**) a projeção horizontal. Uma vez que a projeção horizontal **P** e a profundidade **z** são impostos pelas características da plataforma e das conexões nos poços, e o ângulo **α** é relacionado à projeção **P** e ao comprimento total (**L1 + L2 + L3**), apenas estes últimos parâmetros geométricos precisam ser considerados no processo da otimização.

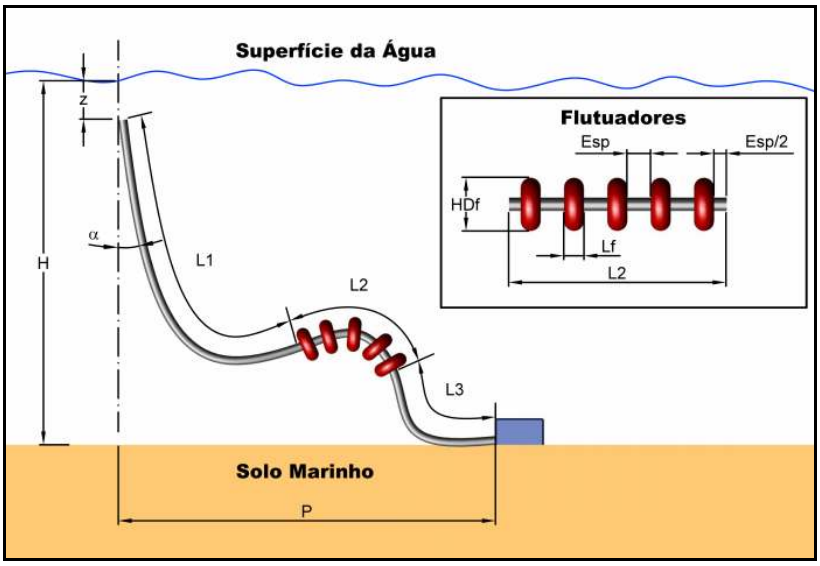

Figura 41. Variáveis de projeto do problema de *risers*.

Há também os parâmetros relativos às bóias (flutuadores), que são (**Lf**), comprimento da bóia; (**HDf**), o diâmetro da bóia, (**Esp**) o espaçando entre as bóias. Outros parâmetros tais como o peso específico e outras propriedades mecânicas das bóias podiam ser considerados; porém neste trabalho somente os parâmetros geométricos **Lf**, **HDf** e **Esp** serão otimizados. Deve-se enfatizar que o número de bóias não está fixado sobre diferentes execuções da ferramenta de otimização, mais exatamente, este número varia de acordo com o comprimento do segmento com bóias (**L2**), o comprimento de cada bóia (**Lf**), e o afastamento entre as bóias (**Esp**). Em resumo, as variáveis principais do projeto no problema de otimização são **L1**, **L2**, **L3**, **HDf**, **Lf** e **Esp**. Os valores das outras variáveis são calculados a partir destes parâmetros de busca ou dados por valores fixos de entrada, como definidos em (VIEIRA, 2008).

6.1.2 Função de Custo

Em problemas de otimização de engenharia estrutural, a função objetivo envolve geralmente o mais baixo custo de construção (associado a uma função de custo), e a conformidade com todos os padrões técnicos e critérios de projeto (associados às restrições de projeto ou às funções de penalidade).

Para a aplicação de *riser* considerada neste trabalho, o comprimento e o custo de cada segmento assim como o custo das bóias, são considerados para construir a função de custo do problema, que é dada por:

$$f = \frac{f_{\min}}{\left(\sum_{i=1}^{n} IC_i \times L_i\right) + \left(V_{flut} \times IC_{flut}\right)} \tag{60}$$

Nesta expressão, $i = 1...n$ representa o índice do segmento do *riser*; IC_i é o índice de custo (por metro) associado a cada segmento; L_i é o comprimento do segmento; IC_{flut} representa o índive de custo associado ao volume da bóia; V_{flut} é o volume da bóia e $f_{\min}$ é o valor mínimo da função objetivo usado para normalizar o valor do custo no intervalo [0, 1].

6.1.3 Restrições de Projeto

As restrições de projeto do problema de otimização são determinadas a partir dos resultados de análises estruturais das configurações do *riser*. Estas restrições estruturais de comportamento são as seguintes:

- A tensão máxima equivalente de *Von Mises* que atua nas seções do *riser* (para assegurar a integridade estrutural da tubulação);
- O ângulo máximo do *riser* com relação ao eixo vertical na conexão com a plataforma (imposto por exigências de instalação);
- A variação máxima do ângulo de "*built-in*", medida no topo do eixo do *riser*, entre a configuração neutra de equilíbrio e alguma configuração adquirida pelo *riser* durante a aplicação dos carregamentos ambientais e dos movimentos da plataforma (impostos pelo projeto da *flex-joint* que fornece uma conexão articulada do *riser* com a plataforma);
- A tensão máxima no topo do *riser* (igualmente imposta pelo projeto da *flex-joint*); e
- A tensão mínima na parte inferior do *riser* (para evitar o encurvamento e o colapso de uma seção do *riser*).

Uma função de penalidade associada à violação de cada uma destas cinco restrições de projeto é definida em termos da razão *x* entre o limite de restrição e o valor calculado, como segue:

$$P = \begin{cases} k \times \left(1 - x^3\right), & \text{if } x < 1 \\ 0, & \text{if } x \geq 1 \end{cases} \tag{61}$$

onde *k* é um fator que permite o surgimento de soluções não restritas.

Neste momento, é interessante notar que a ferramenta *ProgOtim* incorpora dois procedimentos diferentes para a avaliação do comportamento estrutural do *riser*: **1)** Um método de resolução baseado na análise dinâmica não-linear no domínio do tempo de Elementos Finitos (EF), e **2)** Um método de resolução analítico da catenária. Naturalmente, é sabido que o uso do modelo analítico aproximado da catenária seria uma forte limitação e impediria que o método fosse usado em aplicações no mundo real. Conseqüentemente, em atividades reais de projeto o usuário deve selecionar o método de resolução por EF.

Entretanto, a fim alcançar o objetivo principal do estudo apresentado neste capítulo (isto é, "calibrar" o algoritmo do PSO definindo os melhores valores e o comportamento da variação de seu conjunto de parâmetros), um número muito grande de experiências deve ser executado. Se tais experiências fossem executadas usando análises completas de EF, um custo computacional muito grande seria exigido.

Portanto, nas experiências apresentadas na seção seguinte, as avaliações são executadas usando o método de resolução da catenária, que é muito mais rápido para computar e fornece resultados que são, pelo menos, representativos da solução real por EF. Esta abordagem foi empregada já com sucesso no trabalho de (VIEIRA, 2008), no contexto dos estudos relativos ao uso de Algoritmos Genéticos. Depois que os parâmetros do método de PSO forem ajustados, em aplicações reais de projeto as avaliações das configurações do *riser* podem ser executadas pelo MEF.

6.1.4 *Fitness* ou Função objetivo

Finalmente, para este problema de minimização com restrição, o objetivo ou a função de *fitness* (ou função de aptidão) é dada por

$$fitness = 100 \times \left[\frac{f}{1.0 + \sum P_j} \right] \tag{62}$$

onde P_j é a penalidade relativa à violação do j-ésimo critério de restrição de projeto. Com esta expressão normalizada, escalada pelo fator (100), a escala de valores possíveis de *fitness* encontra-se no intervalo [0, 100].

6.2 Experimentos no algoritmo PSO aplicado ao projeto dos *risers*

6.2.1 Descrição do estudo de caso

Os estudos nas variações do PSO descritas neste trabalho serão executados para um *riser* em configuração *lazy-wave*, para ser instalado em uma profundidade de mar de 1290m, e considerando uma projeção horizontal de 2000m. O comportamento das variações será avaliado em termos da *fitness* da solução obtida.

Dados específicos relacionados à modelagem do *riser* são descritos na Tabela 1. Nesta tabela, a taxa de custo C2/C1 significa que o segmento com flutuadores custa duas vezes mais do que os segmentos regulares do *riser*. Os limites definidos pelo usuário para as variáveis de projeto são especificados na Tabela 2. Os limites para as restrições estruturais de comportamento do *riser* são apresentados na Tabela 3.

Tabela 1. Dados do *Riser*.

Material	
Densidade	7800 kg/m³
Peso Específico	77 kN/m³
Tensão de Escoamento	413 MPa
Tensão Admissível	277 MPa
Módulo de Elasticidade	207800 MPa
Taxa de custo C2/C1	2,0

Geometria	
Espessura	0,01905 m
Diâmetro Externo	0,21908 m
Diâmetro Interno	0,18098 m
Peso do Flutuador	0,162 ton/m
Empuxo do Flutuador	0,3175 ton/m
Diâmetro externo do flutuador	0,568 m

Tabela 2. Limites das variáveis de projeto.

Variável	**Min (m)**	**Max (m)**
Segmento do *Riser* (L1) – topo	800	2000
Segmento do *Riser* (L2)	400	800
Segmento do *Riser* (L3) – fundo	800	2000
Diâmetro do flutuador (HDf)	0,8	2
Comprimento do flutuador (Lf)	0,5	2
Espaçamento entre os flutuadores (Esp)	1,0	3,0

Tabela 3. Restrições de projeto.

Restrição	**Valor**
Tensão de *Von Mises*	415,5 MPa
Ângulo de topo máximo	18°
Ângulo de topo mínimo	5°
Variação do ângulo de "built-in"	5°
Tração máxima no topo	1500 kN
Tração mínima	300 kN

6.2.2 Descrição dos Experimentos

Esta seção caracteriza os experimentos executados com as variações do PSO descritas neste trabalho, a fim de estudar o comportamento da convergência do algoritmo na otimização de *risers laze-wave*.

Baseado em resultados anteriores apresentados em (ALBRECHT, 2005), todos os testes consideraram uma população de 10 partículas. Tais resultados indicaram que o espaço de busca não é explorado suficientemente com um número menor de partículas, e o algoritmo mostra convergência prematura. Por outro lado, com mais de 10 partículas os tempos de computação são aumentados sem melhorar significativamente os resultados.

Cada experimento corresponde a valores selecionados para os parâmetros descritos na seção 3.6, a saber:

- O coeficiente de inércia, em termos de um valor fixo ω, ou de um valor inicial ω_0 para a variação linear ou não-linear (de acordo com as Equações (7) e (9) respectivamente); e
- Os valores iniciais e finais que definem a variação linear dos coeficientes de agregação e de congregação C_1, C_2 e C_3 (C_{1ini} , C_{1fin} , C_{2ini} , C_{2fin} , C_{3ini} , C_{3fin} de acordo com as Equações (10), (11) e (12)).

A Tabela 4 define oito conjuntos de valores para os valores iniciais e finais para os coeficientes C_1, C_2 e C_3, aumentando ou diminuindo em uma escala entre 1 e 2. Para cada conjunto da Tabela 4, seis valores diferentes para o valor inicial ω_0 do coeficiente de inércia ω são considerados: 0.4, 0.8, 1.2, 1.6, 2.0 e 2.4, com os três tipos de comportamento diferentes: fixo ($\omega = \omega_0$), ou com variação linear ou não-linear de acordo com as Equações (7) e (9), respectivamente. Para a variação não-linear o valor do expoente *n* na Eq. (9) foi tomado como igual a 1.2.

Em resumo, cada experimento corresponde a uma combinação de um conjunto de parâmetros da Tabela 4 com os seis valores diferentes para o coeficiente de inércia inicial ω_0 e três tipos diferentes de comportamento para ω – fixo, com variação linear ou com variação não-linear. O algoritmo foi executado 30 vezes para cada experimento, cada execução correspondendo a uma realização dos termos aleatórios do método.

Tabela 4. Conjuntos de valores iniciais e finais dos coeficiente C_1, C_2 e C_3.

Conjunto	C_1		C_2		C_3	
	Inicial	**Final**	**Inicial**	**Final**	**Inicial**	**Final**
1	1	2	1	2	1	2
2	1	2	1	2	2	1
3	1	2	2	1	1	2
4	1	2	2	1	2	1
5	2	1	1	2	1	2
6	2	1	1	2	2	1
7	2	1	2	1	1	2
8	2	1	2	1	2	1

A execução do algoritmo é concluída quando o valor médio de *fitness* permanece acima de 98% do melhor valor de *fitness* encontrado ao longo de cinco iterações consecutivas, significando que as partículas estão concentradas perto de uma solução ótima.

A fim de comparar os resultados com aqueles obtidos utilizando outras abordagens, o desempenho do algoritmo PSO no problema de otimização proposto foi avaliado com o uso dos parâmetros fixos propostos em (TRELEA, 2003). Com esta finalidade, foram realizadas execuções adicionais empregando os dois conjuntos de parâmetros fixos apresentados em (TRELEA, 2003):

a) $\omega = 0{,}6$; $C_1 = C_2 = 1{,}7$, e **b)** $\omega = 0{,}729$; $C_1 = C_2 = 1{,}494$.

6.2.3 Resultados

Os resultados estatísticos das realizações dos experimentos são apresentados nas Tabelas 5 a 7, em termos de média (μ) e desvio padrão (σ) dos valores de *fitness*. Estas tabelas correspondem, respectivamente, aos experimentos com variação fixa, linear e não-linear do coeficiente de inércia ω. Obviamente, o bom desempenho do algoritmo é medido por valores médios de *fitness* maiores, e desvios padrão menores.

A fim de permitir uma avaliação mais fácil destes resultados, são resumidos os melhores resultados nas Tabelas 8 e 9, respectivamente, para cada conjunto de coeficientes C_1, C_2 e C_3 da Tabela 4, e para cada valor de inércia inicial ω_0. Finalmente, os resultados são indicados graficamente nas Figuras 42 a 45.

Tabela 5. Resultados dos experimentos com inércia fixa.

Conj.	*Fitness*											
	ω = 0,4		**ω = 0,8**		**ω = 1,2**		**ω = 1,6**		**ω = 2,0**		**ω = 2,4**	
	μ	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**
1	49,12	5,355	61,37	0,174	60,88	0,580	60,43	0,949	59,60	0,876	59,77	0,937
2	50,42	5,957	60,49	2,408	60,62	0,950	60,40	1,161	60,15	0,642	59,65	0,891
3	53,46	4,613	61,24	0,641	60,97	0,345	59,88	1,676	59,59	0,933	59,28	0,891
4	59,31	3,806	61,33	0,497	60,81	0,572	60,39	0,641	60,17	0,621	59,43	1,145
5	55,78	5,877	61,11	1,189	60,84	0,470	60,19	1,333	59,74	1,646	58,69	2,224
6	59,72	3,272	61,48	0,098	60,51	1,484	60,26	1,365	59,78	0,973	59,29	1,309
7	59,90	3,170	61,36	0,440	60,75	0,448	60,02	1,690	59,37	1,263	59,44	0,915
8	60,96	1,613	61,49	0,106	60,70	1,216	60,20	1,413	59,58	1,281	59,28	1,411

Tabela 6. Resultados dos experimentos, inércia com variação linear.

Conj.	*Fitness*											
	ω_0 = 0,4		**ω_0 = 0,8**		**ω_0 = 1,2**		**ω_0 = 1,6**		**ω_0 = 2,0**		**ω_0 = 2,4**	
	μ	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**
1	49,22	5,654	57,96	4,781	60,70	2,041	61,27	0,550	61,16	0,468	61,17	0,200
2	52,66	5,281	59,47	3,320	61,20	0,859	60,89	1,988	61,20	0,710	61,06	1,237
3	51,47	3,969	59,71	4,306	61,31	1,127	61,30	1,126	61,13	1,162	61,23	0,557
4	58,22	3,562	60,98	1,358	61,52	0,101	61,53	0,112	61,46	0,127	61,30	1,235
5	56,93	4,917	60,97	1,852	61,48	0,515	60,94	1,567	61,12	0,792	61,27	0,583
6	60,47	2,381	60,94	1,466	60,59	3,422	61,46	0,432	61,18	1,175	60,83	1,530
7	59,63	3,661	61,19	1,254	61,39	0,624	61,44	0,210	61,09	1,399	61,42	0,161
8	61,09	1,070	60,86	1,400	61,37	0,604	61,45	0,171	61,45	0,102	61,23	0,653

Tabela 7. Resultados dos experimentos, inércia com variação não-linear (n = 1,2).

Conj.	*Fitness*											
	ω_0 = 0,4		**ω_0 = 0,8**		**ω_0 = 1,2**		**ω_0 = 1,6**		**ω_0 = 2,0**		**ω_0 = 2,4**	
	μ	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**
1	47,73	4,015	60,03	2,890	61,14	1,139	61,28	0,276	61,26	0,257	61,03	0,523
2	52,86	5,458	59,79	3,902	61,33	1,135	61,43	0,130	61,37	0,489	61,20	1,108
3	51,25	4,500	61,27	0,658	61,40	0,534	61,44	0,159	61,18	0,736	61,26	0,540
4	58,64	3,291	61,03	1,502	61,50	0,218	61,54	0,108	61,45	0,177	61,44	0,304
5	56,23	5,368	60,32	3,282	61,51	0,105	60,72	1,707	61,36	0,158	61,03	1,209
6	59,29	3,905	60,82	1,634	61,22	1,032	61,41	0,551	61,47	0,107	61,08	1,221
7	59,13	3,562	61,37	0,698	61,23	1,063	61,48	0,133	60,98	1,535	61,22	0,643
8	61,46	0,439	61,53	0,088	61,39	0,536	61,12	1,000	61,12	1,083	61,03	1,129

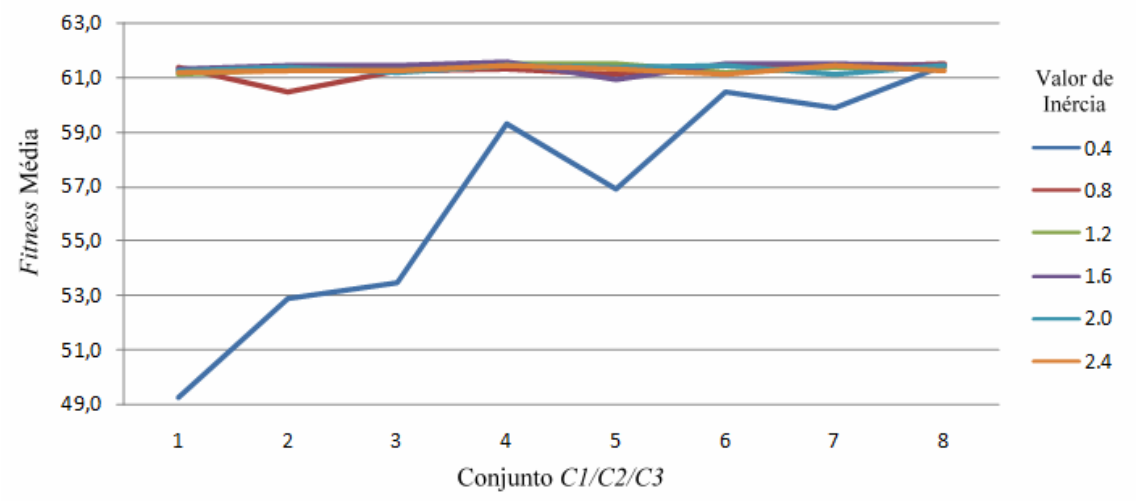

Figura 42. *Fitness* (μ) x ($C_1/C_2/C_3$), para diferentes valores iniciais de inércia.

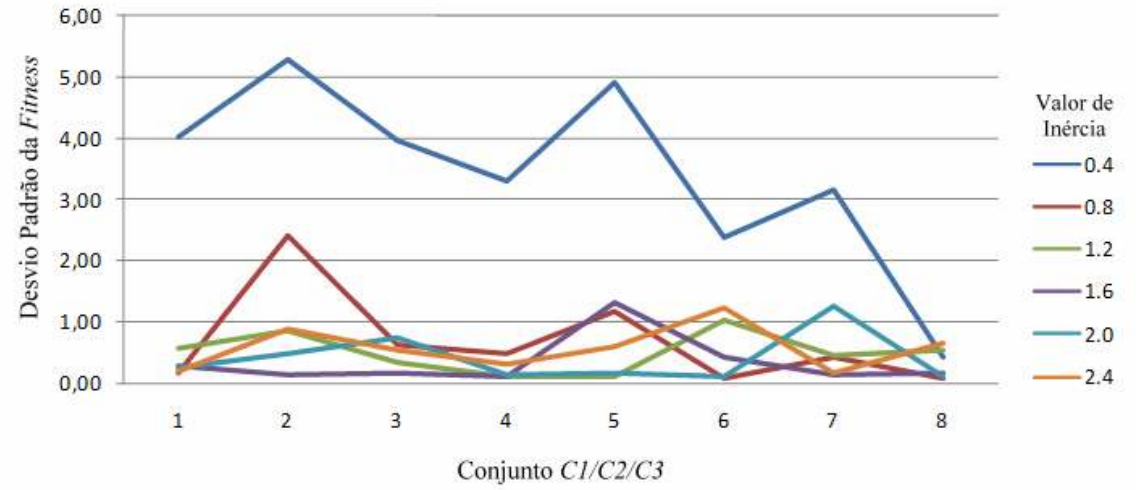

Figura 43. *Fitness* (σ) x ($C_1/C_2/C_3$), para diferentes valores iniciais de inércia.

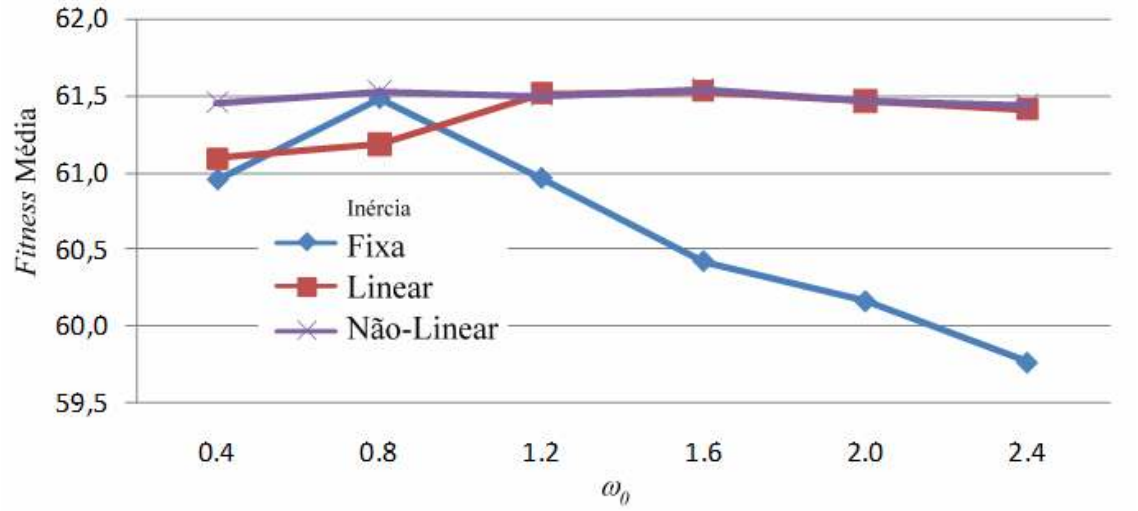

Figura 44. *Fitness* (μ) x ω_0, para diferentes tipos de variação da inércia.

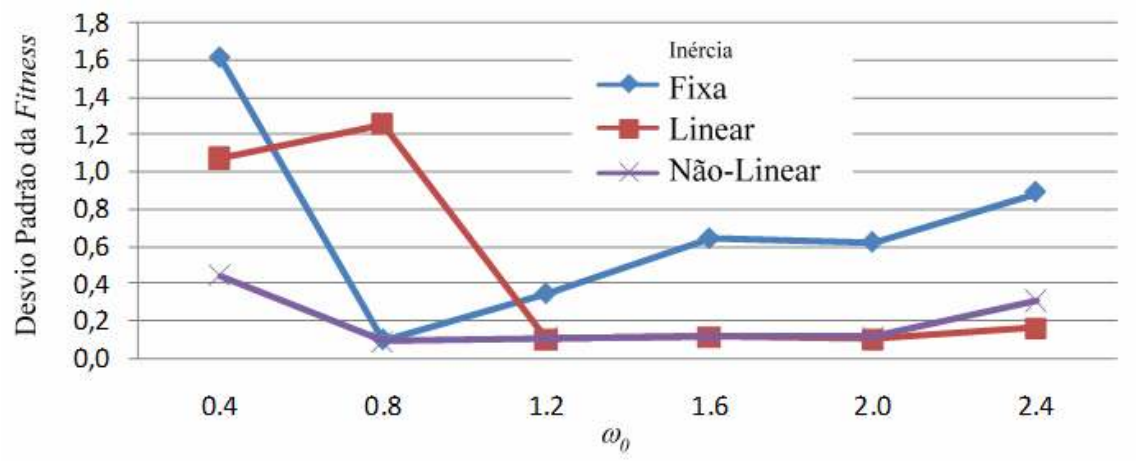

Figura 45. *Fitness* (σ) x ω_0, para diferentes tipos de variação da inércia.

A respeito da influência do coeficiente da inércia ω, nas Figuras 42 e 43 pode-se observar que o desempenho do algoritmo PSO é afetado significativamente pelo uso de valores menores para ω. Este comportamento é mais perceptível para o valor fixo ω = 0,4 (ou o valor inicial ω_0 = 0,4 nos casos com variação linear ou não-linear), mas pode ser igualmente observado para alguns casos com o valor ω = 0,8 (ou ω_0 = 0,8). Nestes casos, o algoritmo alcança uma convergência prematura para configurações de *riser* com valores médios de *fitness* menores (como indicado na Figura 42); também, os resultados apresentam desvios padrão mais elevados (Figura 43). Por exemplo, como indicado na Tabela 7, o menor valor médio de *fitness* (47,73) obtido entre todos os testes corresponde ao Conjunto 1 e ω_0 = 0,4.

As Figuras 42 e 43 também indicam que, para valores de coeficientes de inércia maiores que ω = 0,8, os resultados não são visivelmente diferentes (embora valores médios de *fitness* ligeiramente mais baixos e desvios padrão mais elevados possam ser observados para ω > 2,0). Isto sugeriria a seleção de ω (ou ω_0) na escala entre 1,2 e 2,0.

Entretanto, enquanto as Figuras 44 e 45 confirmam que os resultados não são visivelmente diferentes para ω_0 > 0,8 com variação linear ou não-linear de ω (novamente com exceção de pequenas diferenças para ω > 2,0), ao usar valores fixos para ω as diferenças são notáveis. Isto pode ser verificado observando a curva azul correspontente aos experimentos com ω fixo, que apresentam, na maioria dos casos, *fitness* médias mais baixas e desvios padrão mais elevados.

Este é o primeiro indicativo de que o uso de um coeficiente ω fixo possa não ser a melhor abordagem para este problema. De fato, observando também as Figuras 44 e 45,

pode-se concluir que, não obstante o valor selecionado para ω ou ω_0, o comportamento do algoritmo é melhor (com *fitness* médias maiores e desvios padrão menores) ao adotar a variação não-linear do coeficiente de inércia ω. Esta conclusão é enfatizada observando-se as Figuras 46 e 47, onde pode-se notar que, independentemente do conjunto de valores considerados para os coeficientes C_1, C_2 e C_3, a variação não-linear de ω conduz a um desempenho melhor.

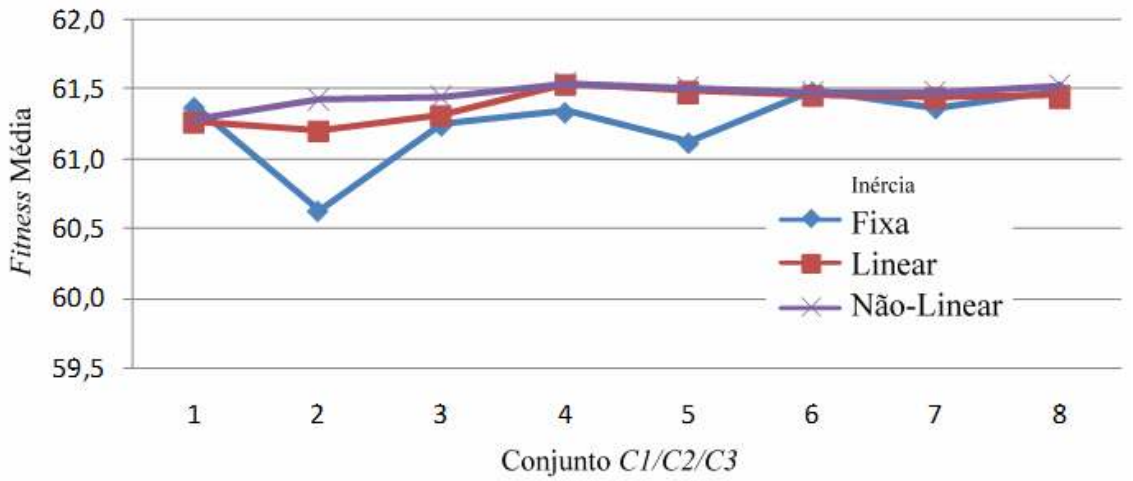

Figura 46. *Fitness* (σ) x (C_1/C_2/C_3), para diferentes tipos de variação da inércia.

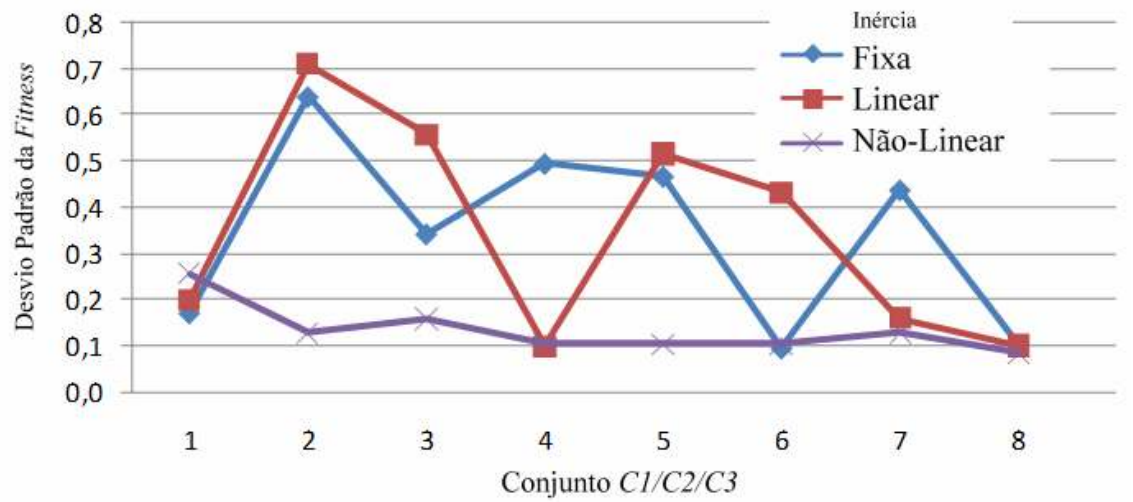

Figura 47. *Fitness* (σ) x (C_1/C_2/C_3), para diferentes tipos de variação da inércia.

As Figuras 42, 43, 46 e 47 indicam que, descartando os casos com coeficiente de inércia ω fixo ou com variação linear, e com menores valores para o peso inercial inicial ω_0, os resultados não são notavelmente influenciados pelo conjunto de C_1, C_2 e C_3. Isto é, para o comportamento recomendado para o coeficiente da inércia ω (com variação

não-linear) e a escala recomendada para ω_0 (entre 1,2 e 2,0), os diferentes conjuntos de coeficientes de agregação e de congregação conduzem a valores médios de *fitness* similares. Porém algumas pequenas diferenças nos valores de desvio padrão podem ser observadas, que, associados às *fitness* médias mais elevadas, indicam que os conjuntos 4 ou 8 da Tabela 4 poderiam ser uma boa escolha.

Como pode ser visto nas Tabelas 8 e 9, entre todos os experimentos executados, o conjunto dos coeficientes $C_1/C_2/C_3$ que conduz ao valor médio de *fitness* mais elevado (61,54) é o de número 4 (com o C_1 variando linearmente de 1 para 2, e C_2/C_3 variando linearmente de 2 para 1), com variação não-linear do coeficiente de inércia ω, e com valor inicial $\omega_0 = 1,6$. Entre as 30 execuções para este experimento, o maior valor individual de *fitness* obtido foi 61,69. Os parâmetros do sistema de *riser* obtido para essa execução são indicados na Tabela 10. A pequena diferença entre este valor máximo de *fitness* e a *fitness* média (somente 0,15, ou 0,24%) é refletida no fato de que este experimento forneceu também um dos valores mais baixos do desvio padrão (0,108) obtidos entre todos os testes.

Tais valores baixos de desvio padrão indicam que o algoritmo é robusto. Isto pode ser considerada uma das características mais importantes do método de otimização, junto com a habilidade de fornecer valores médios mais elevados de *fitness*. De fato, quando comparamos estes resultados com aqueles obtidos usando os parâmetros fixos propostos em (TRELEA, 2003), apresentados na Tabela 11, pode-se observar que, embora a *fitness* média relatada acima não seja muito maior do que a obtida com os parâmetros fixos (60,99 e 60,87), o desvio padrão é muito mais baixo do que o obtido com os parâmetros fixos de (TRELEA, 2003) (1,258 e 1,885).

Tabela 8. Melhores resultados, para cada conjunto C_1, C_2 e C_3.

Conjunto C_1, C_2 C_3	**ω Fixo**		**ω Linear**		**ω Não-linear**	
	μ	**σ**	**μ**	**σ**	**μ**	**σ**
1	61,37	0,174	61,27	2,182	61,28	0,257
2	60,62	0,642	61,20	2,063	61,43	0,130
3	61,24	0,345	61,31	1,882	61,44	0,159
4	61,33	0,497	61,53	1,103	**61,54**	**0,108**
5	61,11	0,470	61,48	2,653	61,51	0,105
6	61,48	0,098	61,46	1,890	61,47	0,107
7	61,36	0,440	61,44	1,125	61,48	0,133
8	61,49	0,106	61,45	1,201	61,53	0,088

Tabela 9. Melhores resultados, para cada valor inicial de inércia.

ω_0	ω Fixo		ω Linear		ω Não-linear	
	μ	σ	μ	σ	μ	σ
0,4	60,96	1,613	61,09	1,070	61,46	0,439
0,8	61,49	0,098	61,19	1,254	61,53	0,088
1,2	60,97	0,345	61,52	0,101	61,51	0,105
1,6	60,43	0,641	61,53	0,112	**61,54**	**0,108**
2,0	60,17	0,621	61,46	0,102	61,47	0,107
2,4	59,77	0,891	61,42	0,161	61,44	0,304

Tabela 10. Valores para os parâmetros da melhor configuração de *riser* encontrada.

Parâmetro	L1	L2	L3	HDf	Lf	Esp
Valor	1473	400	800	1,70	1,10	1,33

Tabela 11. Resultados para os coeficientes fixos de acordo com (TRELEA, 2003).

C_1	C_2	ω	μ	σ
1,7	1,7	0,6	60,99	1,258
1,494	1,494	0,729	60,87	1,885

6.2.4 Desempenho Computacional

Finalmente, considerando o desempenho computacional, os custos computacionais do próprio algoritmo de otimização podem ser considerados irrelevantes para o tipo de aplicação considerado aqui. Os custos são concentrados no método que será empregado para avaliar o desempenho de cada configuração do *riser* em aplicações reais de projeto: um método de resolução baseado na análise dinâmica não-linear no domínio do tempo de Elementos Finitos. Consequentemente, os tempos de computação do algoritmo de otimização devem ser expressados em termos do número de avaliações do *riser* que são executadas.

Ao longo da execução dos estudos paramétricos, observou-se que o número de avaliações aumenta com o valor do coeficiente de inércia ω. Para o menor valor fixo $\omega = 0{,}4$, o algoritmo necessitou de aproximadamente 400 avaliações. Com os valores fixos "ótimos" propostos por (TRELEA, 2003), $\omega = 0{,}6$ e $\omega = 0{,}729$, um número médio de, respectivamente, 570 e 735 avaliações foram exigidos. Para valores mais elevados para ω fixo o número de avaliações é aumentado consideravelmente, e o algoritmo pára na maioria dos casos depois que o limite de 1000 avaliações é alcançado.

Por outro lado, observou-se que o número de avaliações é reduzido com o uso da variação linear ou não-linear do coeficiente da inércia ω. Ao usar o valor inicial ω_0 em uma escala entre 0,4 e 1,2, foram necessárias 200 a 500 avaliações do *riser*; para valores mais elevados o número de avaliações não excedeu 900.

Em resumo, considerando somente a eficiência e a estabilidade do algoritmo (em termos da *fitness* média e do desvio padrão), a conclusão geral dos experimentos executados sugere que um comportamento mais estável do PSO, com *fitness* médias mais elevadas e desvios padrão mais baixos, possa ser obtido usando a variação não-linear do coeficiente de inércia ω, associada a ω_0 em uma escala entre 1,2 e 2,0, e os conjuntos 4 ou 8 dos parâmetros C_1, C_2 e C_3. Quando o desempenho computacional é considerado, recordando que o número de avaliações do *riser* aumenta com ω_0, o valor recomendado para ω_0 estreita-se para abaixo de 1,2.

6.3 Conclusões

A definição de configurações eficientes para os *risers* de produção de petróleo conectados às plataformas flutuantes é um assunto crítico para o setor petroleiro. Soluções de custos possíveis são exigidas para a exploração de óleo com o aumento da profundidade das águas, nos cenários onde o custo e a complexidade das estruturas envolvidas igualmente tendem a aumentar - consequentemente motivando estudos sobre procedimentos de otimização.

Neste contexto, foram apresentados neste capítulo estudos na aplicação do método de PSO para a definição de uma configuração eficiente de um *riser* rígido em catenária *lazy-wave*. As variações do método básico de PSO foram consideradas, as quais envolvem a variação de seus parâmetros ao longo das iterações do algoritmo, incluindo os coeficiente de agregação e de congregação passiva, e o coeficiente de peso da inércia.

Os estudos paramétricos extensivos foram executados usando tais variações, onde cada experimento consistiu em uma combinação dada dos valores fixos ou iniciais dos parâmetros, e o tipo de variação. Para cada experimento, foram feitas diversas execuções com valores diferentes para os termos aleatórios do método, coletando os resultados em termos de valores estatísticos de média e desvio padrão do valor de *fitness* da configuração do *riser* obtida. Uma vez que trabalhos anteriores indicaram que o comportamento do método PSO é dependente dos valores selecionados para seus parâmetros, o objetivo deste estudo era determinar um conjunto de parâmetros que conduzem a um desempenho e estabilidade melhores do algoritmo para a determinação

de uma configuração ótima do *riser*, associado com os mais baixos custos computacionais.

Os resultados indicaram que uma customização eficaz do método PSO para o projeto de *risers* rígidos em catenária *lazy-wave* poderia ser obtida adotando uma variação não-linear do coeficiente da inércia ω dado pela Equação (9), com o expoente $n = 1,2$ e o valor inicial $\omega_0 = 1,2$. Embora a influência da variação dos coeficientes de agregação/congregação C_1, C_2 e C_3 venha a ser menos significativa, pelo menos para a escala recomendada para ω_0, os melhores resultados (com desvios padrão menores) foram obtidos com o uso dos conjuntos 4 ou 8 destes parâmetros.

Um outro aspecto relativo às vantagens de qualquer método de otimização, e o método de PSO em particular, são relacionados ao número de parâmetros que devem ser ajustados manualmente pelo usuário. Seguindo este pensamento, em (TRELEA, 2003) somente dois parâmetros fixos foram propostos (ω e $C_1 = C_2$). Poder-se-ia discutir que a abordagem apresentada aqui aumenta significativamente o número de parâmetros a serem ajustados: ω , C_1, C_2 , C_3 e o tipo de variação (linear, não-linear, e os respectivos valores iniciais e finais). Entretanto, o aumento no número de parâmetros reduz somente a facilidade na execução do método, mas não atrapalha seu desempenho computacional. Depois que o método foi implementado na ferramenta Prog*Otim* (com os melhores valores e comportamentos da variação para os parâmetros incorporados no código), pode-se considerar que o objetivo desta análise foi atingido, isto é, uma ferramenta de otimização "amigável" que ajusta automaticamente a maioria dos parâmetros relevantes, customizada para o projeto de *risers*.

7 METAMODELOS ASSOCIADOS À ANÁLISE DINÂMICA

Embora o uso de métodos de otimização torne mais fácil o projeto de sistemas *offshore*, evitando uma análise paramétrica manual exaustiva, a análise dinâmica do modelo de elementos finitos ainda deve ser aplicada em cada etapa do algoritmo de otimização.

Os resultados de uma análise dinâmica no domínio do tempo consistem na série temporal dos parâmetros de resposta de interesse. É importante ressaltar que, para alguns projetos de engenharia, pode ser necessária a obtenção de uma série temporal suficientemente longa, o que faz com que cada uma das análises necessárias exijam um tempo computacional elevado. Este é o caso da análise dinâmica de estruturas *offshore*, que necessita que a série temporal dos parâmetros de resposta (esforços, tensões, movimentos, etc) seja suficientemente longa, a fim de conseguir estabilidade estatística nos resultados obtidos na análise extrema, assim como na análise de fadiga.

Geralmente, quando há dificuldades em estabelecer ou aplicar um modelo matemático para investigar um problema físico, modelos substitutos (metamodelos) mais simples são usados para descrever a resposta dinâmica do sistema (CASSINI & AGUIRRE, 1999), o que permite a predição da resposta de sistemas dinâmicos em função somente dos dados de entrada, não exigindo o conhecimento prévio de parâmetros dinâmicos do sistema (PINA & ZAVERUCHA, 2008).

As Redes Neurais Artificiais (RNA) ganharam especial atenção na previsão de séries temporais devido à sua habilidade de aprendizado e sua capacidade de generalização, associação e busca paralela. Estas qualidades as tornam capazes de identificar e assimilar as características mais marcantes das séries, tais como sazonalidade, periodicidade, tendências, entre outras, na maioria das vezes camufladas por ruídos, sem necessitar do trabalhoso passo de formulação teórica, imprescindível em procedimentos estatísticos (ABELÉM, 1994).

Além disso, as RNAs também têm se destacado pelos seguintes aspectos: sua capacidade em lidar com dados não lineares; sua robustez, pois são capazes de se auto corrigir mesmo depois de previsões erradas e; sua facilidade de utilização. É interessante ressaltar que, apesar de oferecer vantagens, as RNAs também possuem problemas. Um dos principais é a falta de procedimentos para definir com precisão o número de camadas intermediárias ou o número de neurônios em cada uma destas camadas, ou seja, a configuração mais apropriada para a aplicação estudada.

Na análise dinâmica de estruturas *offshore*, GUARIZE (2004) e MATOS (2005) verificaram a possibilidade de aplicação de redes neurais artificiais, obtendo bons resultados. Em (GUARIZE *et al.*, 2007), foi apresentado um procedimento híbrido eficiente, usando redes neurais e elementos finitos, 20 vezes mais rápido do que uma simulação completa usando somente elementos finitos. Tais análises consideraram a utilização de redes neurais como modelos com entradas exógenas, ou seja, modelos onde o valor atual da série temporal de resposta é relacionado apenas aos valores atuais e passados das séries de entrada. As entradas exógenas são as séries que ajudam a prever a variável de interesse.

Sendo assim, propõe-se, no presente trabalho, a utilização de redes neurais como modelos exógenos autoregressivos não-lineares (*nonlinear autoregressive exogenous* - NARX), isto é, modelos autoregressivos não-lineares que tenham entradas exógenas. Isto significa que os modelos relacionam o valor atual da série temporal a valores passados da mesma série, valores atuais e passados das séries exógenas e um termo residual.

Neste trabalho foi implementado em MATLAB um modelo de rede neural, bem como um código de criação dos dados de análise a partir de séries temporais, sendo possível variar o número de atrasos considerados em cada série do problema. Os dados de análise foram separados em conjuntos de treinamento, validação e teste. Para efeito de comparação, foram criados exemplos baseados nos dois modelos citados: o modelo só com entradas exógenas e o modelo NARX. A comparação entre os dois modelos foi feita tomando-se como parâmetro o erro médio quadrático obtido nas predições.

A seguir, serão apresentados dois exemplos de aplicação de redes neurais artificiais como metamodelos para análise dinâmica de estruturas. O primeiro exemplo é um sistema dinâmico simples (massa-mola-amortecedor) com apenas um grau de liberdade e rigidez linear. Esse exemplo foi utilizado apenas por motivos acadêmicos de forma a verificar a aplicação prática das redes neurais, fazendo uma comparação dos dois modelos de criação de dados de análise, bem como investigar a sensibilidade das mesmas em relação aos atrasos temporais considerados para a resposta e ao número de exemplos necessários para o treinamento.

O segundo exemplo considerado envolve a avaliação dos modelos de predição aplicados na análise de linhas de ancoragem conectadas a um FPSO com sistema de ancoragem do tipo DICAS, projetado para uma lâmina d'água de 2000m.

A rede neural artificial utilizada em todos os experimentos de ambos os exemplos foi uma *feedforward multilayer perceptron*, com 10 neurônios na camada intermediária e com função de ativação tangente hiperbólica. Todos os dados de entrada e saída foram normalizados dentro do intervalo [-1,1]. O número de entradas varia dependendo do problema e do modelo testado.

Os pesos da rede foram inicializados com valores aleatórios e o algoritmo de treinamento usado foi o *Levenberg-Marquardt backpropagation* (HAGAN & MENHAJ, 1994). Este algoritmo é considerado o método mais rápido para redes neurais *feedfoward* de tamanho moderado (backpropagation com gradiente descendente é, em geral, muito lento para problemas práticos).

7.1 Sistema dinâmico massa-mola-amortecedor

Como exemplo de um sistema dinâmico simples, temos o sistema massa-mola-amortecedor com um grau de liberdade apresentado na Figura 48. Este mesmo sistema foi utilizado em (MATOS, 2005) e, com uma formulação diferente, em (GUARIZE, 2004).

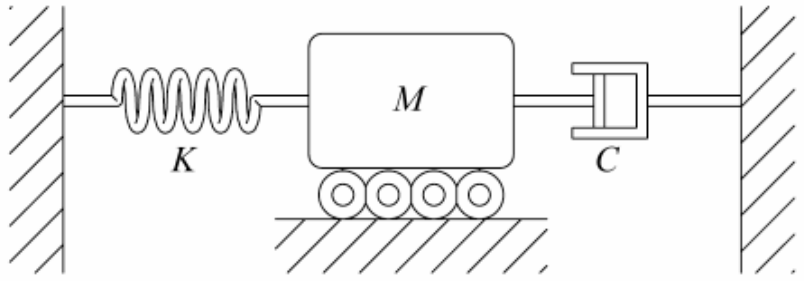

Figura 48. Sistema massa-mola-amortecedor

O sistema é submetido a um carregamento $f(t)$ dado pela equação:

$$f(t) = a\eta(t) \tag{63}$$

onde $\eta(t)$ é uma função de elevação de onda considerando um espectro de Pierson-Moskowitz gerado para um estado de mar com altura significativa de onda (H_s) igual a 7,8m, período de cruzamento zero (T_z) igual a 11,8s, e ângulo de incidência 0 (zero) graus com o eixo *x*. O gráfico do espectro de Pierson-Moskowitz pode ser visto na Figura 49. Na equação (63), *a* é o parâmetro de proporcionalidade que será definido como em (MATOS, 2005).

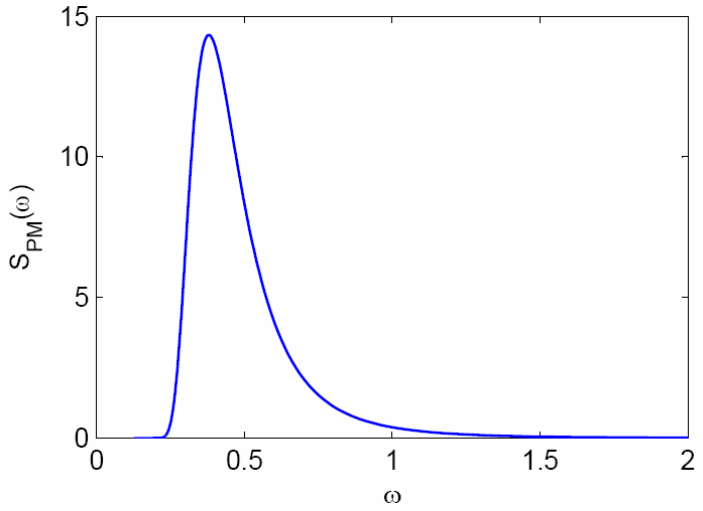

Figura 49. Espectro de Pierson-Moskowitz com $H_s = 7{,}8$m e $T_z = 11{,}8$s.

Considerando a rigidez linear, a equação de equilíbrio do sistema pode ser dada por:

$$x''(t) + 2\xi\omega_n x'(t) + \omega_n{}^2 x(t) = \frac{a}{M}\eta(t) = a_n\eta(t) \tag{64}$$

onde $x(t)$, $x'(t)$, $x''(t)$ são, respectivamente, o deslocamento, a velocidade e a aceleração do sistema, $\omega_n = \sqrt{\frac{K}{M}}$ é a freqüência natural do sistema, sendo K a rigidez linear da mola e M a massa do sistema, ξ é a proporção do amortecimento C em relação ao amortecimento crítico do sistema $C_c = 2\sqrt{M.K}$.

Assim como em (MATOS, 2005), para se amplificar a resposta dinâmica do sistema, foram utilizados os valores $\omega_n = 0.5$ e $\xi = 0.05$ de tal forma que o pico da função de transferência do sistema (CLOUGH & PENZIEN, 1975), fique dentro da região de maior energia das ondas (vide Figura 49). O valor do parâmetro a_n foi o mesmo utilizado por MATOS (2005), que o calculou de forma que o desvio padrão de $x(t)$ ficasse unitário, resultando em $a_n = 0{,}031317$.

A resposta correta do sistema foi calculada com o algoritmo Runge-Kuta de quarta ordem (função ode45 do Matlab) com um intervalo de tempo discreto de 0,5s ($\Delta t = 0.5s$) para um tempo total de simulação de 10800s (3 h).

Para o cálculo de cada caso foi considerada como entrada do sistema a força de excitação $f(t)$ e como saída o deslocamento $x(t)$. Uma vez que os parâmetros variáveis do sistema foram estabelecidos para $x(t)$, eles foram mantidos para avaliar os demais

parâmetros de resposta do sistema: velocidade ($x'(t)$), e aceleração ($x''(t)$). A Figura 50 mostra um trecho da série temporal de força de excitação. As Figuras 51, 52 e 53 mostram trechos das séries temporais dos parâmetros de resposta do sistema (deslocamento, velocidade e aceleração, respectivamente).

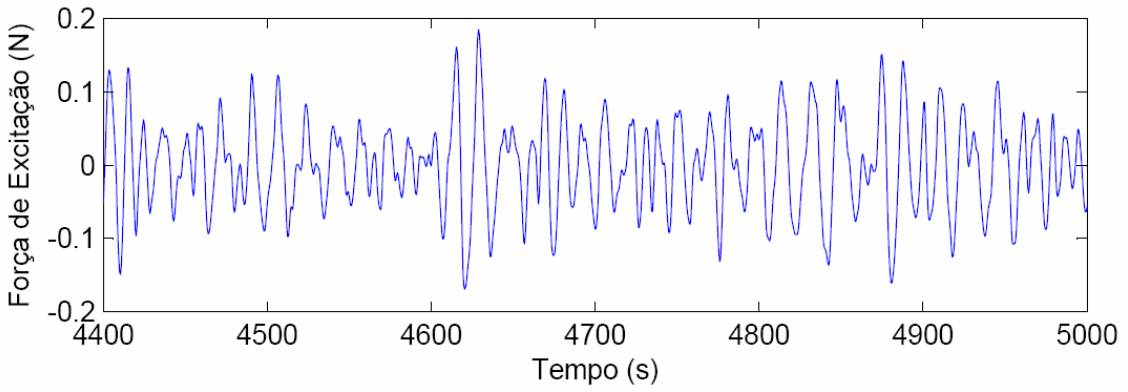

Figura 50. Trecho da série temporal de força de excitação.

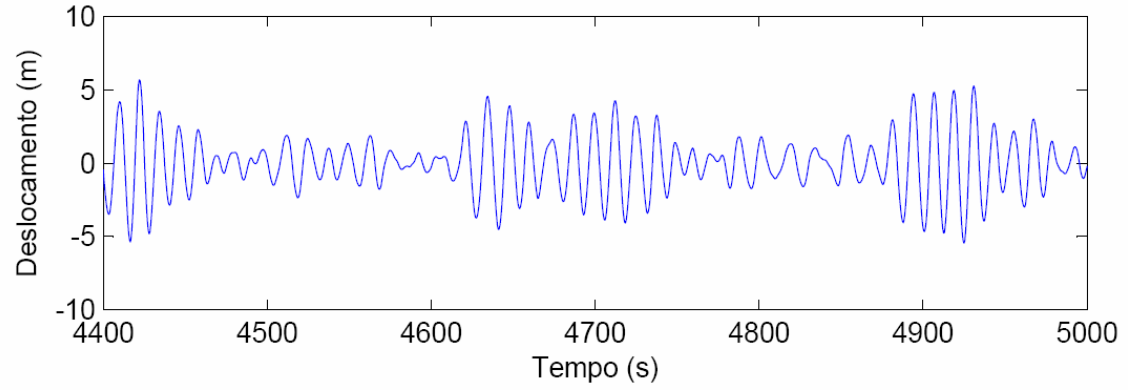

Figura 51. Trecho da série temporal de deslocamento.

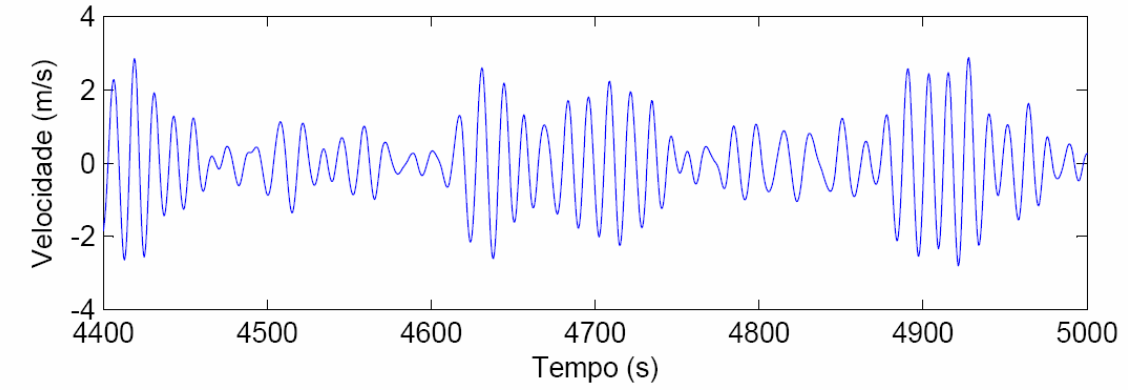

Figura 52. Trecho da série temporal de velocidade.

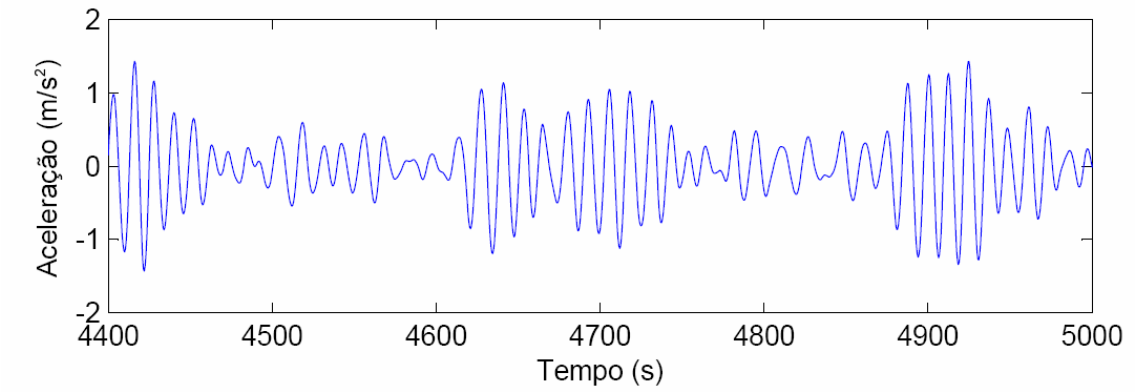

Figura 53. Trecho da série temporal de aceleração.

Pretende-se que, a partir de um curto período de simulação, seja possível fazer a predição do comportamento do sistema no tempo restante utilizando uma rede neural artificial como metamodelo para o sistema massa-mola-amortecedor.

Como explicado anteriomente, utilizou-se dois modelos diferentes para a criação da rede reural. O primeiro usa apenas a série temporal da força de excitação como entrada para a rede neural artificial:

$$r(t) = f(x(t), x(t-\Delta t), \ldots, x(t-N_x \Delta t)) \tag{65}$$

onde $r(.)$ é a resposta (deslocamento, aceleração ou velocidade) estimada pela rede neural; $x(.)$ é a força de excitação; N_x é o atraso (número de pontos precedentes usados na predição) para a força de excitação; e Δt é a discretização do tempo. MATOS (2005) utilizou este modelo, fazendo a predição dos parâmetros de resposta do sistema através dos modelos NARMAX (polinomial e racional) e por Redes Neurais Artificiais. Este modelo será referido como FEX (Força de Excitação).

O segundo método, usa uma rede neural como um modelo NARX. Neste exemplo, a variável de interesse é o deslocamento (e, em seguida, velocidade e aceleração), a entrada exógena é a série de força de excitação, e o termo residual foi ajustado para zero:

$$r(t) = f(r(t-\Delta t), \ldots, r(t-N_r \Delta t), x(t), x(t-\Delta t), \ldots, x(t-N_x \Delta t)) \tag{66}$$

onde $r(.)$, $x(.)$, N_x e Δt são definidos como anteriormente, e N_r é o número de pontos precedentes da série temporal de deslocamento (velocidade ou aceleração) usados na predição da resposta atual. Este modelo será referido como NARX.

A Tabela 12 descreve cada um dos casos testados, 6 deles usando o modelo FEX e 6 usando o modelo NARX. Os modelos FEX não recebem dados precedentes da série

de resposta (deslocamento/velocidade/aceleração) como entrada. Cada atraso corresponde a um intervalo de tempo $\Delta t = 0.5s$.

Tabela 12. Casos testados para o problema massa-mola-amortecedor.

Método	Caso	N_x	N_r
FEX	1	140	-
	2	150	-
	3	160	-
	4	170	-
	5	180	-
	6	190	-
NARX	1	140	15
	2	140	30
	3	150	15
	4	150	30
	5	160	15
	6	160	30

Foram utilizadas amostras de treinamento de 300s e 400s, com conjuntos de validação de 75s e 100s, respectivamente. Os 50s iniciais foram descartados como uma fase transiente. Assim, de acordo com o tempo total de simulação testado (10800s), serão utilizados, nos casos com 300s e 400s, respectivamente, conjuntos de teste com 10375s e 10250s, equivalendo a 20750 e 20500 exemplos a serem testados na rede treinada.

O treinamento da rede foi realizado visando um erro médio quadrático mínimo no conjunto de treinamento de 10^{-5} e considerando um número máximo de épocas igual 200. Além disso, se o erro no conjunto de validação aumentar por 6 épocas seguidas, o treinamento é concluído.

Para cada caso estudado foram realizadas 20 execuções independentes. A Tabela 13 mostra os resultados obtidos para cada caso estudado (calculada a média e o desvio padrão sobre as 20 execuções independentes) na predição da série de deslocamento. Todos os resultados estão indicados graficamente na Figura 54.

Pode-se observar na Figura 54 que o aumento do conjunto de treinamento influencia mais o modelo NARX do que o modelo FEX, mostrando uma melhora significativa dos resultados quando são inseridos mais 100s no treinamento.

Analisando a Tabela 13 e a Figura 54, pode-se observar que o menor erro médio no conjunto de teste obtido com o modelo FEX foi no caso 6 (com 190 atrasos na força de excitação) utilizando 400s de treinamento.

Tabela 13. Resultados experimentais das predições para a série de deslocamento.

Método	Caso	Amostra de Treino (s)	Erro de Treinamento		Erro de Teste		Épocas	Tempo (s)
			μ	σ	μ	σ		
FEX	1	300	3,706e-05	1,092e-05	1,240e-02	7,000e-04	20	44,128
		400	1,193e-04	1,047e-04	1,204e-02	2,727e-03	35	71,619
	2	300	6,347e-05	5,551e-05	1,086e-02	1,547e-03	22	54,034
		400	5,982e-05	1,380e-05	9,487e-03	1,275e-03	33	76,890
	3	300	1,091e-04	1,163e-04	9,171e-03	1,574e-03	27	74,294
		400	3,750e-05	1,024e-05	8,860e-03	1,795e-03	47	129,159
	4	300	2,701e-05	8,327e-06	7,454e-03	2,819e-03	13	39,816
		400	2,616e-04	4,815e-04	6,126e-03	1,862e-03	32	96,716
	5	300	2,942e-05	2,541e-05	7,020e-03	1,255e-03	17	55,316
		400	3,430e-05	1,891e-05	5,780e-03	1,158e-03	40	137,547
	6	300	2,040e-05	8,847e-06	6,442e-03	1,629e-03	15	54,478
		400	1,783e-04	3,275e-04	**5,134e-03**	1,458e-03	34	131,459
NARX	1	300	1,103e-05	3,280e-06	6,103e-03	2,191e-03	17	40,494
		400	9,475e-06	8,792e-07	2,274e-03	2,330e-04	21	58,331
	2	300	1,301e-05	4,437e-06	4,672e-03	1,905e-03	18	55,981
		400	1,330e-05	4,866e-06	**2,077e-03**	1,681e-04	17	57,328
	3	300	1,056e-05	8,065e-07	5,012e-03	1,033e-03	13	34,025
		400	9,993e-06	1,090e-06	3,028e-03	2,130e-04	29	91,284
	4	300	1,258e-05	3,340e-06	4,120e-03	1,886e-03	16	52,100
		400	8,112e-06	1,828e-06	2,110e-03	2,414e-04	15	58,665
	5	300	1,068e-05	2,357e-06	4,325e-03	7,896e-04	13	40,863
		400	1,352e-05	1,309e-05	2,845e-03	8,765e-04	20	75,037
	6	300	1,008e-05	2,468e-06	3,733e-03	3,842e-04	10	38,625
		400	8,192e-06	3,422e-06	2,131e-03	2,299e-04	13	54,193

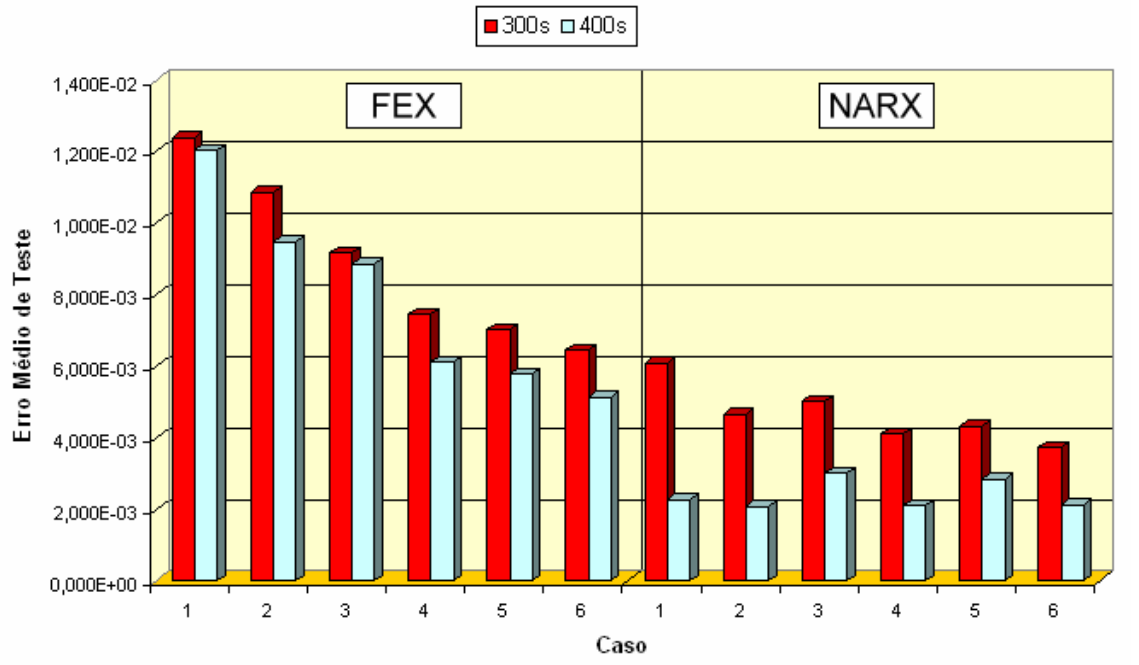

Figura 54. Erros médios de predição dos modelos FEX e NARX.

Entre os modelos NARX, pode-se ver que o caso 2 (com 140 atrasos na força de excitação e 30 atrasos na série de deslocamento) utilizando 400s de treinamento obteve os melhores resultados, conseguindo os mais baixos valores de média e desvio padrão do erro de teste. Na Tabela 13, os resultados de média e desvio padrão na coluna "Erro de Treinamento" engloba os erros nos pontos de treinamento e de validação. Comparando o melhor modelo FEX (caso 6) com o melhor modelo NARX (caso 2), ambos utilizando 400s para treinamento da rede e 100s para validação, pode-se notar que o modelo NARX supera o modelo FEX em todos os aspectos.

A Figura 55 mostra um intervalo de tempo com duas séries sobrepostas: a série de resposta (deslocamento) calculada pelo algoritmo Runge-Kuta no sistema massa-mola-amortecedor (MMA), e as predições feitas pelo modelo FEX no caso 6 em uma das 20 execuções realizadas utilizando 400s de treinamento. Tal intervalo de tempo foi escolhido porque apresentou o maior erro médio quadrático nos pontos previstos.

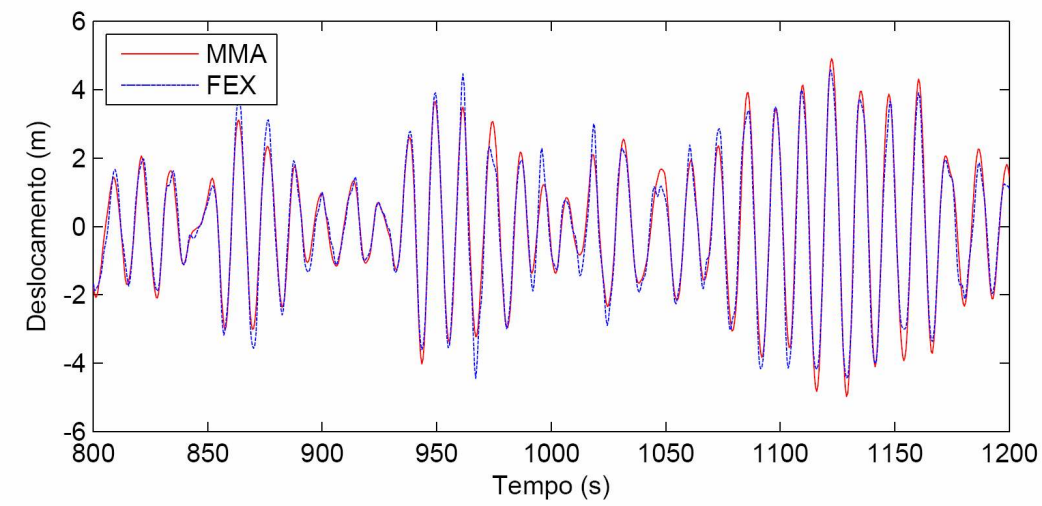

Figura 55. Predições da série de deslocamento obtidas com o modelo FEX.

Analogamente, a Figura 56 mostra a série de resposta (deslocamento) calculada pelo algoritmo Runge-Kuta, e as predições feitas pelo modelo NARX no caso 2 utilizando 400s de treinamento. Pode-se verificar que o modelo NARX fornece predições mais exatas do que o modelo FEX. Isto pode ser visualmente confirmado analisando a Figura 57, que mostra somente os erros de predição de ambos os modelos nas mesmas circunstâncias.

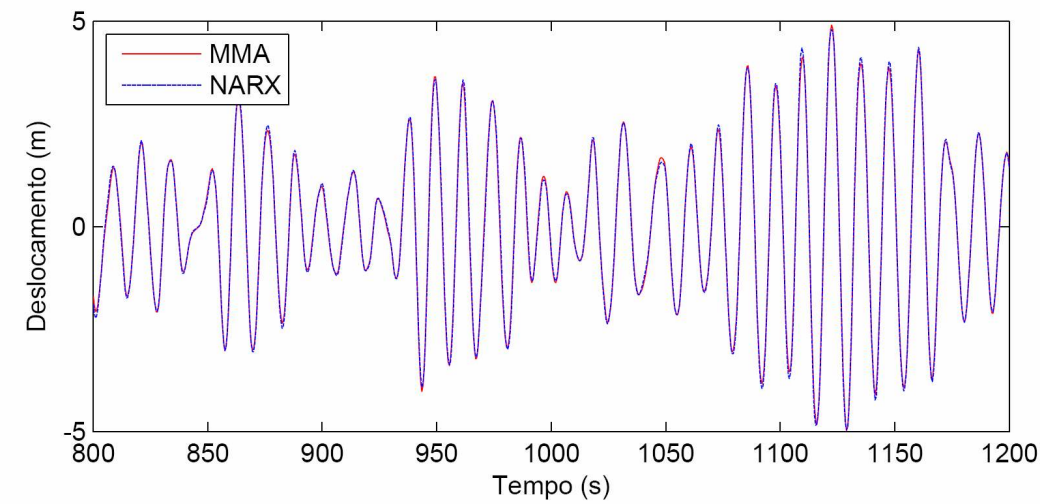

Figura 56. Predições da série de deslocamento obtidas com o modelo NARX.

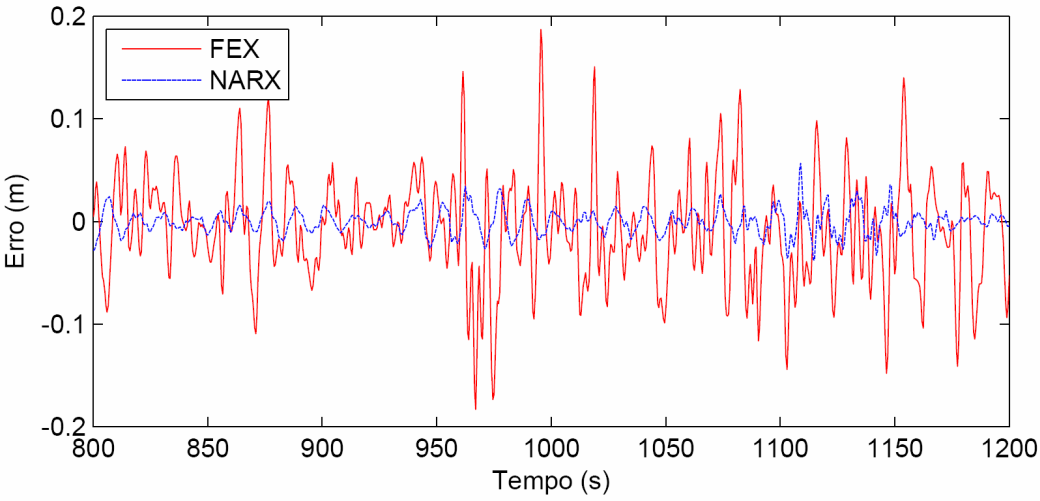

Figura 57. (FEX x NARX): Erros de predição na série de deslocamento.

Utilizando as duas configurações para os modelos FEX e NARX escolhidas a partir de seus desempenhos na predição da série de deslocamento, foram realizadas mais 20 execuções independentes de treinamento de redes neurais para cada um dos demais parâmetros de resposta do sistema massa-mola-amortecedor (velocidade e aceleração). Os resultados obtidos na predição das séries de velocidade e aceleração podem ser vistos nas Tabelas 14 e 15, respectivamente.

Tabela 14. Resultados experimentais da predição da série de velocidade.

Amostra de Treino	**Método**	**Caso**	**Erro de Treinamento**		**Erro de Teste**		**Épocas**	**Tempo (s)**
			μ	σ	μ	σ		
400s	FEX	6	4,748e-04	9,746e-04	6,823e-03	3,137e-03	31	135,981
	NARX	2	1,155e-05	3,261e-06	**2,207e-03**	4,110e-04	30	100,350

Tabela 15. Resultados experimentais da predição da série de aceleração.

Amostra de Treino	Método	Caso	Erro de Treinamento		Erro de Teste		Épocas	Tempo (s)
			μ	σ	μ	σ		
400s	FEX	6	1,437e-04	2,672e-04	5,312e-03	1,373e-03	38	160,587
	NARX	2	1,106e-05	2,733e-06	**1,868e-03**	1,547e-04	24	81,975

Analisando as Tabelas 14 e 15, respectivamente, pode-se observar que o modelo NARX escolhido também supera o modelo FEX na predição das séries de velocidade e aceleração em todos as aspectos.

Analogamente às Figuras 55, 56 e 57, as Figuras 58, 59 e 60 mostram os resultados das predições para a série de velocidade e as Figuras 61, 62 e 63 os resultados para a série de aceleração.

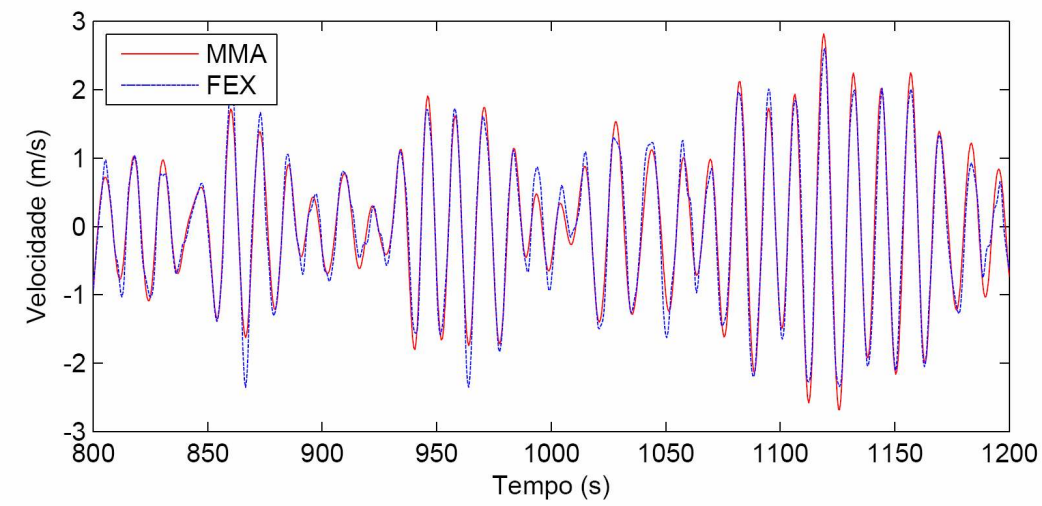

Figura 58. Predições da série de velocidade obtidas com o modelo FEX.

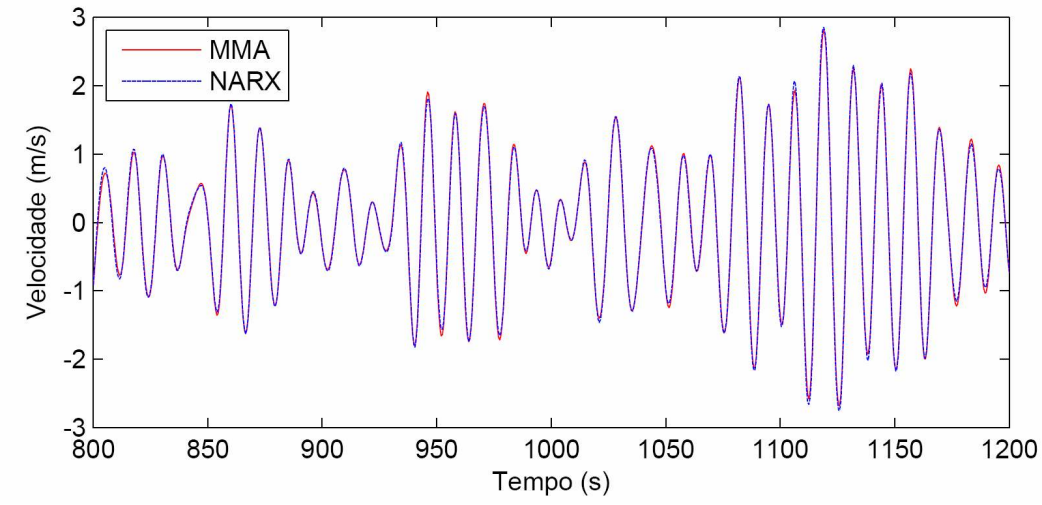

Figura 59. Predições da série de velocidade obtidas com o modelo NARX.

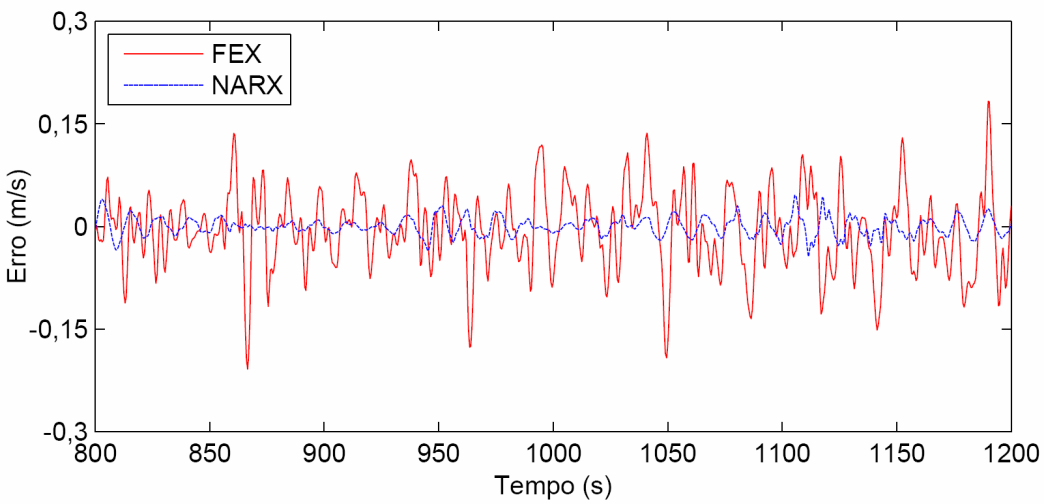

Figura 60. (FEX x NARX): Erros de predição na série de velocidade.

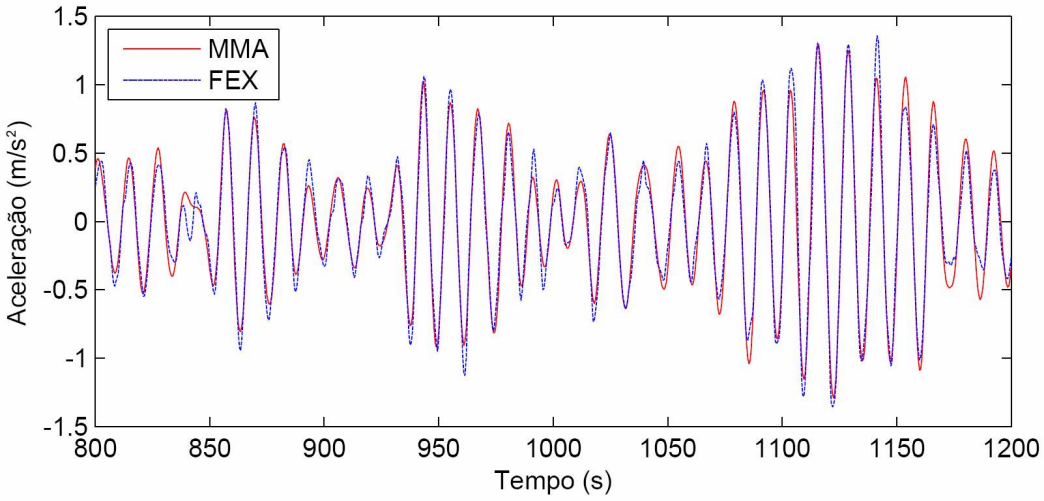

Figura 61. Predições da série de aceleração obtidas com o modelo FEX.

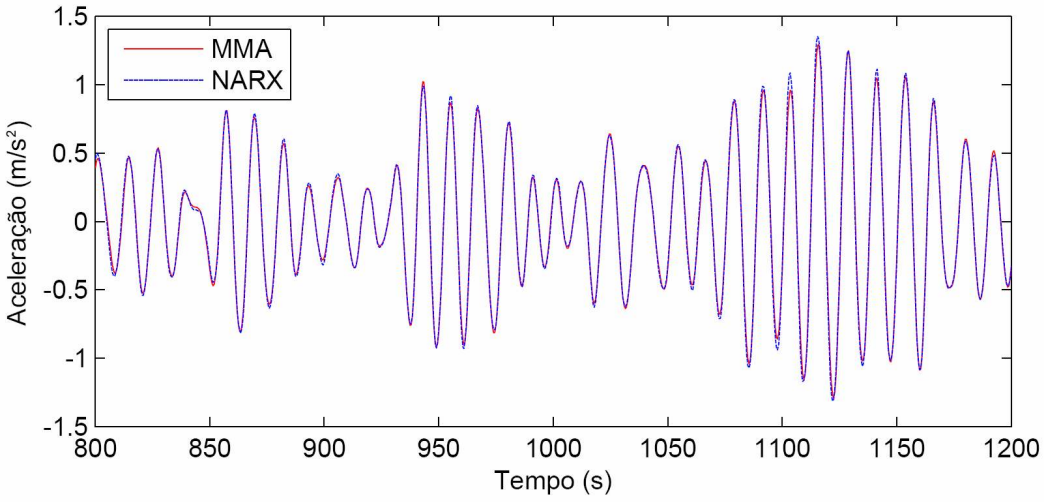

Figura 62. Predições da série de aceleração obtidas com o modelo NARX.

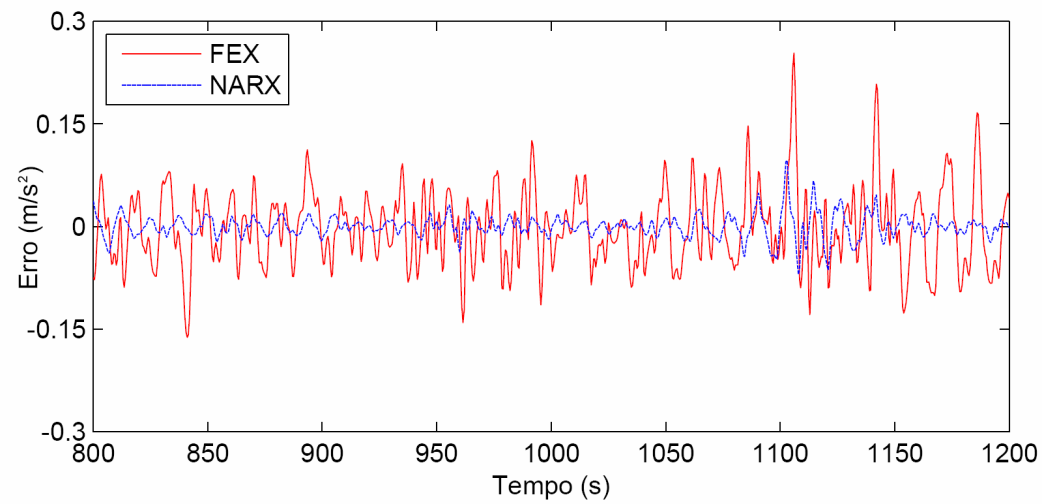

Figura 63. (FEX x NARX): Erros de predição na série de aceleração.

Como uma última observação, deve-se notar que o modelo NARX no caso 2 conseguiu melhores resultados do que o modelo FEX no caso 6 com uma rede neural menor. A rede treinada no modelo FEX no caso 6 possui 191 entradas (190 pontos precedentes da força de excitação mais 1 ponto atual da força de excitação - vide Equação (65)) totalizando 1920 pesos a serem ajustados na rede, enquanto que a rede treinada no modelo NARX no caso 2 possui 171 entradas (140 pontos precedentes da força de excitação, 30 pontos precedentes da série de resposta, mais 1 ponto atual da força de excitação - vide Equação (66)) totalizando 1720 pesos a serem ajustados. O número de entradas afeta diretamente o tempo de execução do treinamento da rede.

7.2 Linhas de Ancoragem

Em uma análise dinâmica de uma linha de ancoragem, a resposta é a tração dinâmica no topo da linha. O método de elementos finitos calcula os movimentos da unidade flutuante a fim de calcular a tração no topo da linha. Os movimentos usados possuem 3 graus de liberdade: *surge* (movimento longitudinal), *sway* (movimento lateral), e *heave* (movimento vertical). As Figuras 64, 65 e 66 mostram trechos das séries de movimento usadas nesta análise (*surge*, *sway* e *heave*, respectivamente). A Figura 67 mostra a série temporal de tração no topo correspondente, calculada pelo método de elementos finitos.

Os dados utilizados nos experimentos são de um sistema flutuante de produção composto por 8 linhas de ancoragem e 5 *risers*, numa profundidade de 2000m. As séries temporais com 10800s de simulação foram obtidas usando carregamentos ambientais como entradas para o programa SITUA/PROSIM.

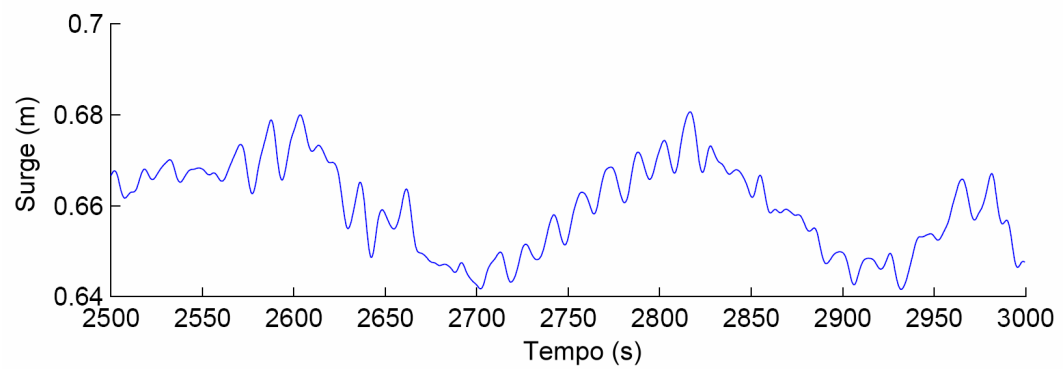

Figura 64. Série de movimento *surge*.

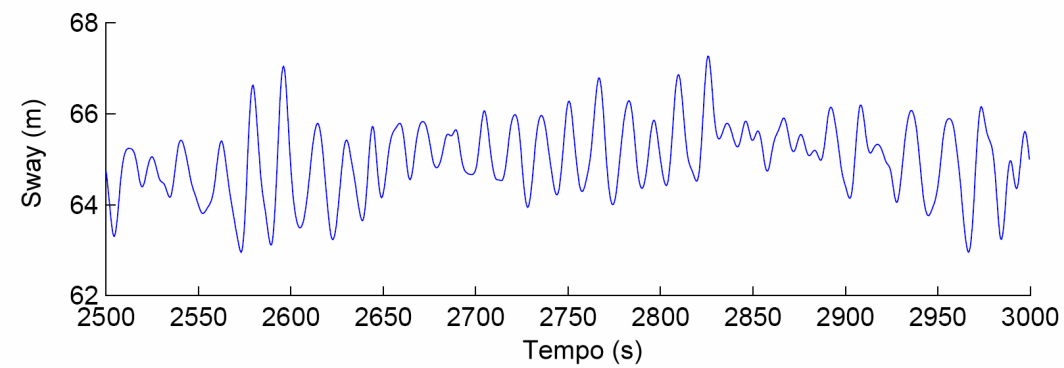

Figura 65. Série de movimento sway.

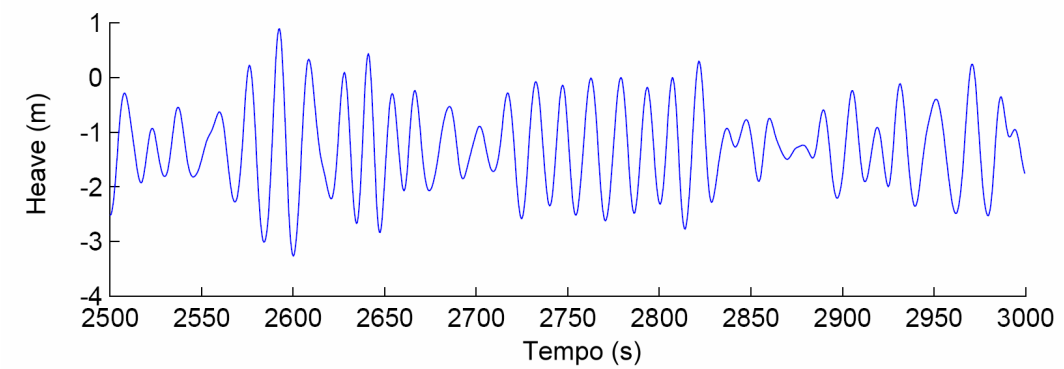

Figura 66. Série de movimento heave.

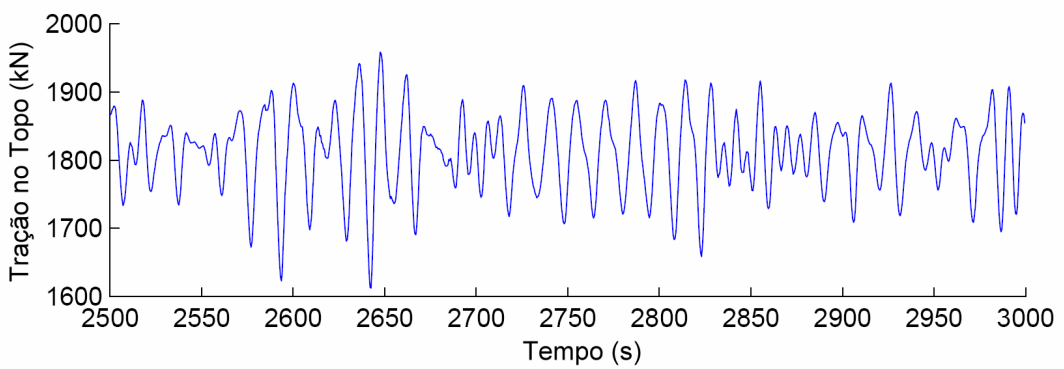

Figura 67. Série temporal de tração no topo da linha.

Pretende-se que, a partir de um trecho curto da série temporal de tração no topo de uma linha de ancoragem, calculada através do método de elementos finitos, seja possível treinar uma rede neural artificial, usando esta série de tração e as séries dos movimentos, capaz de prever os valores futuros de tração no topo.

Como no exemplo anterior, foram utilizados dois modelos diferentes para a criação da rede reural. O primeiro usa as séries dos movimentos como entradas para a rede neural artificial:

$$\begin{aligned} r(t) = f(&x(t), x(t-\Delta t), \ldots, x(t-N_x\Delta t), y(t), y(t-\Delta t), \ldots, y(t-N_y\Delta t), \\ &z(t), z(t-\Delta t), \ldots, z(t-N_z\Delta t)) \end{aligned} \tag{67}$$

onde $r(.)$ é a resposta (tração no topo) estimada pela rede neural; $x(.)$, $y(.)$, e $z(.)$ são os movimentos *surge*, *sway* e *heave*, respectivamente; N_x, N_y e N_z são os atrasos (número de pontos precedentes usados na predição) para os movimentos *surge*, *sway* e *heave*, respectivamente; e o Δt é a discretização do tempo. Este modelo foi usado por GUARIZE *et al.* (2007) para estimar a tração no topo de linhas de ancoragem e será referido como PMI (*prescribed motion inputs* - entradas dos movimentos prescritos).

O segundo método, usa uma rede neural como um modelo NARX, onde a variável de interesse é a tração no topo da linha de ancoragem, as entradas exógenas são as séries dos movimentos (*surge*, *sway* e *heave*), e o termo residual foi ajustado para zero:

$$\begin{aligned} r(t) = f(&r(t-\Delta t), \ldots, r(t-N_r\Delta t), x(t), x(t-\Delta t), \ldots, x(t-N_x\Delta t), \\ &y(t), y(t-\Delta t), \ldots, y(t-N_y\Delta t), z(t), z(t-\Delta t), \ldots, z(t-N_z\Delta t)) \end{aligned} \tag{68}$$

onde $r(.)$, $x(.)$, $y(.)$, $z(.)$, N_x, N_y, N_z e Δt são definidos como anteriormente, e N_r é o número de pontos precedentes da série temporal de tração no topo usados na predição da resposta atual. Novamente, este modelo será referido como NARX.

Testes preliminares com um modelo autoregressivo não-linear puro mostraram resultados ruins devido à não-linearidade do sistema e, portanto, a utilização de tal método foi descartada.

A Tabela 16 descreve cada uma das configurações testadas, 3 delas usando o modelo PMI e 6 usando o modelo NARX. Cada configuração consiste em valores típicos dos atrasos usados para cada uma das séries temporais de entrada. Os modelos PMI não recebem dados precedentes da série de tração no topo como entrada. O atraso para o movimento de *heave* é sempre igual ou maior do que os atrasos para os movimentos de *surge* e *sway*, porque a série do movimento de *heave* tem geralmente

uma freqüência mais elevada. Para atrasos na tração no topo maiores que 10, ocorre uma elevada degradação no desempenho dos modelos NARX.

Tabela 16. Configurações testadas no problema de linhas de ancoragem.

Configuração	N_x	N_y	N_z	N_r
PMI-1	10	10	10	-
PMI-2	10	10	20	-
PMI-3	20	20	20	-
NARX-1	10	10	10	5
NARX-2	10	10	20	5
NARX-3	20	20	20	5
NARX-4	10	10	10	10
NARX-5	10	10	20	10
NARX-6	20	20	20	10

Cada série temporal corresponde a 3h de simulação de uma linha de ancoragem (10800s). Os 200s iniciais de cada série foram descartados como uma fase transiente. Os 10600s restantes foram divididos entre os conjuntos de treinamento, validação e teste. Para os conjuntos de treinamento e validação foram usados 500s, 80% para o treinamento das rede neural (400s) e 20% para a validação dos modelos (100s), para evitar *overfitting*. O conjunto de teste, composto pelos últimos 10100s da série, foi usado para avaliar os modelos, comparando as predições da rede com os valores calculados através do método de elementos finitos. A discretização do tempo da série é de 0,5s, conseqüentemente os conjuntos de treinamento, validação e teste possuem 800, 200, e 20200 pontos, respectivamente.

O treinamento da rede foi realizado visando um erro médio quadrático mínimo no conjunto de treinamento de 10^{-5}. O número máximo de épocas foi ajustado para 300, entretanto, todos os testes pararam mais cedo, porque o algoritmo de treinamento foi ajustado para parar se o erro de validação aumentar por 6 épocas seguidas.

Para cada linha e cada modelo, 20 execuções independentes foram executadas. Os modelos alcançaram resultados semelhantes para todas as 8 linhas de ancoragem. A Tabela 17 mostra os resultados para a linha de ancoragem com maior carregamento (calculada a média sobre as 20 execuções independentes).

Analisando os valores da Tabela 17, pode-se observar que o melhor modelo PMI foi o PMI-2, com 10 atrasos para *surge* e *sway*, e 20 atrasos para *heave*. Além de apresentar os menores valores para a média e o desvio padrão do erro de teste, e o menor número de épocas necessário para o treinamento, o tempo de execução foi muito

similar ao mais baixo obtido com o modelo PMI.

Entre os modelos NARX, pode-se ver que NARX-1 obteve claramente os melhores resultados, com 10 atrasos para todas as séries de movimento, e 5 atrasos para a série de tração no topo. O modelo conseguiu os mais baixos valores de média e desvio padrão do erro de teste, o mais baixo número de épocas, e o mais baixo tempo de execução. Comparando o melhor modelo PMI (PMI-2) com o melhor modelo NARX (NARX-1), pode-se notar que o modelo NARX supera o modelo PMI em todos os aspectos (exceto o desvio padrão do erro do teste).

Tabela 17. Resultados experimentais das predições para a série de tração no topo.

Configuração	Erro de Treinamento		Erro de Teste		Épocas	Tempo (s)
	μ	σ	μ	σ		
PMI-1	3,470e-04	1,444e-04	2,809e-03	7,450e-04	122	24,024
PMI-2	**3,069e-04**	**3,519e-04**	**1,951e-03**	**5,911e-04**	**89**	**24,628**
PMI-3	1,535e-04	4,814e-05	3,155e-03	3,382e-03	101	49,948
NARX-1	**5,154e-05**	**1,451e-05**	**8,236e-04**	**7,087e-04**	**73**	**17,684**
NARX-2	5,880e-05	2,916e-05	9,779e-04	8,802e-04	76	25,273
NARX-3	9,902e-05	1,888e-04	4,408e-03	1,581e-02	74	46,138
NARX-4	4,821e-05	2,894e-05	2,604e-03	6,558e-03	93	25,720
NARX-5	3,822e-05	2,067e-05	8,777e-04	7,406e-04	73	27,683
NARX-6	4,752e-05	2,527e-05	1,175e-03	1,003e-03	77	49,022

A Figura 68 mostra um intervalo de tempo com duas séries sobrepostas: a série de resposta (tração no topo) calculada pelo MEF, e as predições feitas pelo modelo PMI-2 na execução que obteve o pior resultado para a linha de ancoragem com o maior carregamento. Foi feita a vizualização dos resultados da execução que obteve o maior erro no conjunto de teste (pior caso) por representar um limite superior para o erro de predição, garantindo, portanto, que os resultados das 19 execuções restantes é melhor ou igual a este apresentado. Tal intervalo de tempo foi escolhido porque apresentou o maior erro médio quadrático nos pontos previstos.

Analogamente, a Figura 69 mostra a série de resposta calculada pelo MEF, e as predições feitas pelo modelo NARX-1. Pode-se verificar que o modelo NARX fornece predições mais exatas do que o modelo PMI. É possível confirmar visualmente este fato analisando a Figura 70, que mostra somente os erros de predição de ambos os modelos nas mesmas circunstâncias.

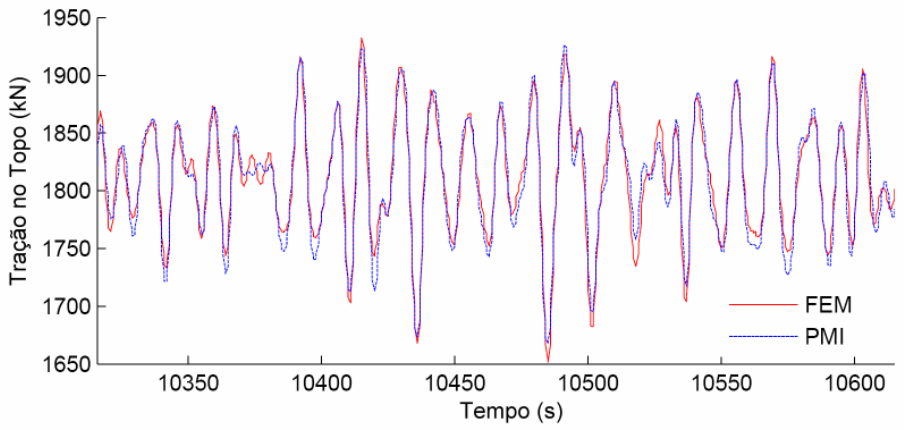

Figura 68. Predições da série de tração no topo obtidas com o modelo PMI-2.

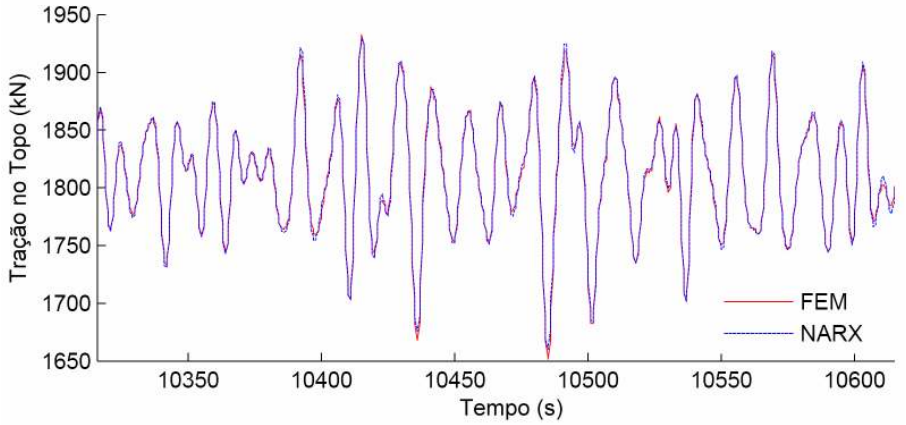

Figura 69. Predições da série de tração no topo obtidas com o modelo NARX-1.

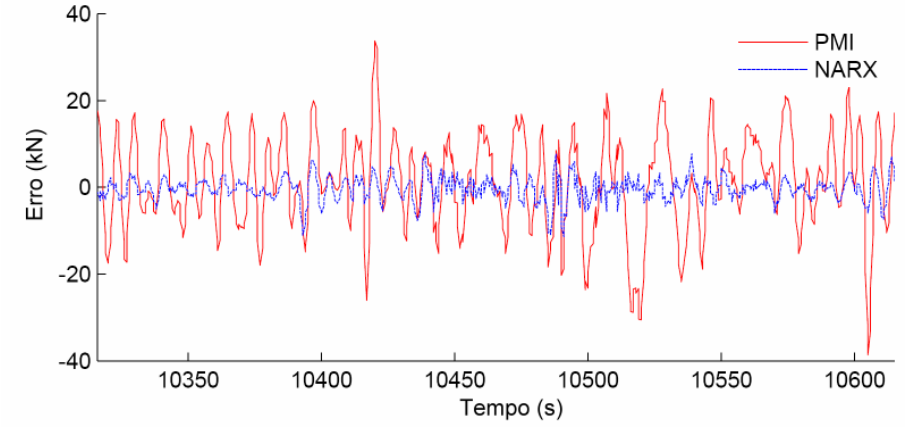

Figura 70. (PMI-2 x NARX-1): Erros de predição na série de tração no topo.

A fim de verificar os resultados encontrados na comparação do erro, foi utilizada a análise REC (*Regression Error Characteristic*) (BI & BENNETT, 2003), uma técnica poderosa para a avaliação e a comparação de modelos de regressão que facilita o visualização do desempenho de muitas funções de regressão simultaneamente em um

único gráfico. As curvas REC generalizam as curvas ROC (PROVOST & FAWCETT, 1997) para regressões com vantagens similares.

Uma curva REC é uma curva monotonicamente crescente que caracteriza a qualidade de um modelo de regressão para níveis diferentes de tolerância do erro. As curvas REC traçam a tolerância do erro no eixo x e a precisão (acurácia) de uma função de regressão no eixo y. A precisão é definida como a porcentagem dos pontos previstos dentro da tolerância.

A área sobre a curva REC (*Area Over Curve* – AOC) é uma estimativa do erro previsto para um modelo de regressão. Quanto menor o AOC, melhor a função de regressão. Uma função de regressão domina outra se sua curva REC está sempre acima da curva REC que corresponde à outra função.

O gráfico REC gerado pode ser visto na Figura 71. Os números entre parênteses na figura são a área sobre os valores da curva para cada curva REC. Analisando os valores AOC e a posição relativa das curvas REC, pode-se concluir que o modelo NARX conseguiu, sem dúvida alguma, o melhor desempenho, dominando o modelo PMI (note que a curva para NARX cobre a curva para PMI) e consequentemente o modelo NARX é preferível.

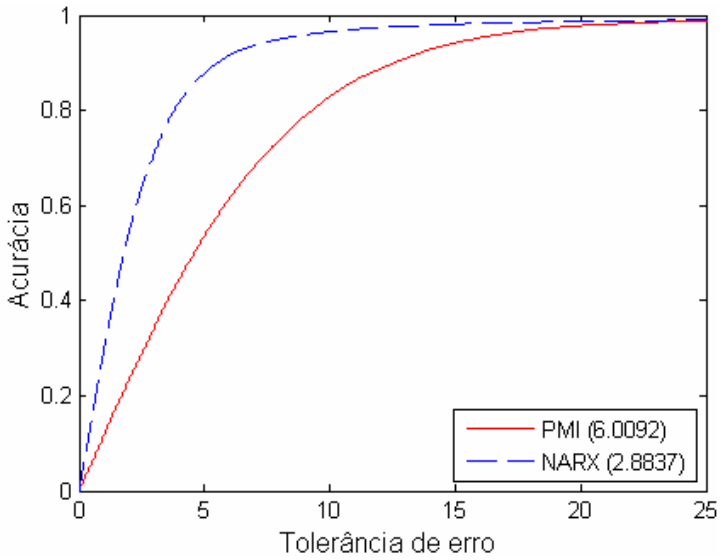

Figura 71. O gráfico REC que compara os modelos NARX-1 e PMI-2.

Como uma última observação, deve-se notar que o modelo NARX-1 conseguiu melhores resultados do que o modelo PMI-2 com uma rede neural menor. A rede neural para o modelo NARX-1 tem 38 entradas (35 pontos precedentes mais 3 pontos atuais do movimento), visto que a rede neural para o modelo PMI-2 tem 43 entradas (40 pontos

precedentes do movimento mais 3 pontos atuais do movimento), e o número de entradas afeta diretamente o número de pesos que devem ser ajustados, e, consequentemente, afetam o tempo de execução.

7.3 Conclusões

Neste capítulo foram apresentados dois exemplos de aplicação de redes neurais artificiais como metamodelos para análises dinâmicas. Os resultados experimentais mostraram que, em ambos os exemplos, a rede neural como um modelo exógeno autoregressivo não-linear (NARX) obteve melhores resultados do que a rede neural como um modelo com entradas exógenas (FEX e PMI).

Na análise do sistema dinâmico massa-mola-amortecedor, os resultados experimentais mostraram que uma rede neural como um modelo NARX foi 56% mais rápida e conseguiu um erro de predição 60% menor do que uma rede neural que usa somente a força de excitação como entrada (FEX).

Da mesma forma, na análise da resposta dinâmica dos sistemas de ancoragem, os resultados experimentais mostraram que uma rede neural como um modelo NARX foi 28% mais rápida e conseguiu um erro de predição 57% menor do que uma rede neural que usa somente os movimentos da unidade flutuante como entradas (PMI), que foi utilizada em pesquisas anteriores (GUARIZE *et al.*, 2007). Os resultados também mostraram que o uso de uma rede neural artificial como um modelo NARX pode fornecer predições de tração no topo com grande precisão, 21 vezes mais rapidamente do que uma simulação completa de 3h usando o método de elementos finitos, uma vez que somente 500s devem ser calculados a fim de construir os conjuntos de treinamento e validação.

O uso de metamodelos na predição de séries temporais de análises dinâmicas se mostrou uma boa ferramenta para diminuir o custo computacional de procedimentos de otimização na busca por configurações ótimas de sistemas de *risers*. Para a criação de tais metamodelos basta tomar como parâmetro de entrada da rede neural os dados de movimentos da embarcação durante todo o tempo de simulação, reduzindo o tempo de análise necessário para obter os parâmetros de saída (esforços, tensões, etc). Esse procedimento deve ser repetido a cada passo do processo de otimização para cada configuração candidata.

Os resultados aqui apresentados foram publicados em *Proceedings of the 2nd International Conference on Engineering Optimization* (PINA *et al.*, 2010b).

8 METAMODELOS ASSOCIADOS À OTIMIZAÇÃO

A fim de reduzir ainda mais o custo computacional e viabilizar o uso de metamodelos em sistemas de ancoragem, onde os movimentos não são dados conhecidos (e sim resultados obtidos de análises de modelos acoplados do casco, das linhas de ancoragem e dos *risers*), pode-se considerar o emprego dos metamodelos de redes neurais diretamente no processo de otimização, utilizando-se como parâmetros de entrada e saída da rede os mesmos parâmetros do algoritmo de otimização. Tomando por base a estratégia de metamodelagem com aproximação sequencial (seção 4.2), torna-se necessário utilizar a análise rigorosa apenas nos primeiros passos do processo de otimização a fim de construir um conjunto de dados para treinamento e validação do metamodelo. Nos passos seguintes da otimização, com a rede neural devidamente treinada, todas as configurações candidatas podem ser avaliadas diretamente através do metamodelo, sem ser necessário o conhecimento do comportamento estrutural inicial dos sistemas.

MENDONÇA (2004) utilizou uma abordagem semelhante aplicando funções aproximadas por redes neurais, treinadas com exemplos obtidos a partir da execução de algoritmos genéticos (AG), na busca do ótimo de funções matemáticas e na otimização da área de uma viga biapoiada com restrições. Já, na área de sistemas *offshore*, MENDONÇA (2004) aplicou uma estratégia híbrida paralela utilizando redes neurais e AG na otimização de movimentos no casco de uma plataforma semi-submersível.

Neste capítulo serão apresentados dois exemplos de aplicação de redes neurais artificiais, treinadas com os indivíduos obtidos nas gerações de processos de otimização por Enxame de Partículas, como metamodelos para aproximação de funções.

O primeiro exemplo de aplicação é simples e baseia-se na utilização de uma rede neural como metamodelo para aproximar a função matemática DeJong F1. Resultados experimentais foram utilizados para avaliar a melhor configuração para a rede neural, bem como o número de indivíduos necessário na população do processo de otimização de forma que a rede neural fosse capaz de aproximar a função DeJong F1 da melhor maneira, ou seja, com o menor erro, sem ser necessário um número muito grande de exemplos de treinamento. Vale ressaltar que esse exemplo é meramente acadêmico, uma vez que o cálculo do valor dessa função não constitui um processo demorado ou computacionalmente custoso. Uma análise semelhante para a aproximação da mesma função utilizando AG pode ser vista em (MENDONÇA, 2004).

No segundo exemplo, será apresentado um estudo para verificar a viabilidade da utilização da estratégia de metamodelagem com aproximação sequencial na otimização de projeto de um sistema de ancoragem composto por 8 linhas de ancoragem e 5 *risers*, numa profundidade de 2400m. Para isso, foram executadas 5 otimizações completas variando o número de indivíduos na população inicial do PSO e utilizando análise estática na avaliação das configurações candidatas. Em seguida, todas as configurações testadas durante esses processos de otimização, foram divididas em dados de treinamento, validação e teste para as redes neurais. Uma vez que o problema de otimização de sistemas de ancoragem retorna três valores de resposta que definem a qualidade da configuração (*offset* máximo, tração no topo máxima e *fitness*), foram propostos três modelos de arquitetura para as redes neurais, considerando-se combinações diferentes dos parâmetros de resposta da otimização como dados de saída para a rede. Em seguida, realizou-se uma análise comparativa dos três métodos propostos. Foi utilizada apenas a análise estática nas avaliações das configurações para poupar custos computacionais, uma vez que a análise dinâmica é muito onerosa. A viabilidade da utilização da rede neural no processo de otimização do sistema analisado foi avaliada por meio do erro médio obtido no conjunto de teste (que abrange as configurações candidatas das últimas gerações da otimização). Depois de comprovada a viabilidade desta aplicação, pode-se realizar avaliações por meio do MEF.

8.1 Função DeJong F1

A função DeJong F1 utilizada é representada pela equação (69). As variáveis da função foram definidas em 3 dimensões (x_1, x_2, x_3) e os valores limites para cada dimensão foram -5,12 e 5,12. A Figura 72 representa o gráfico da função DeJong F1 em 3 dimensões (x_1, x_2, $f(x_1, x_2,0)$) no domínio de variáveis entre -5,12 e 5,12.

$$F_1(x_1, x_2, x_3) = \sum_{i=1}^{3} x_i^{\,2} \qquad -5.12 \le x_i \le 5.12, \;\; 1 \le i \le 3 \tag{69}$$

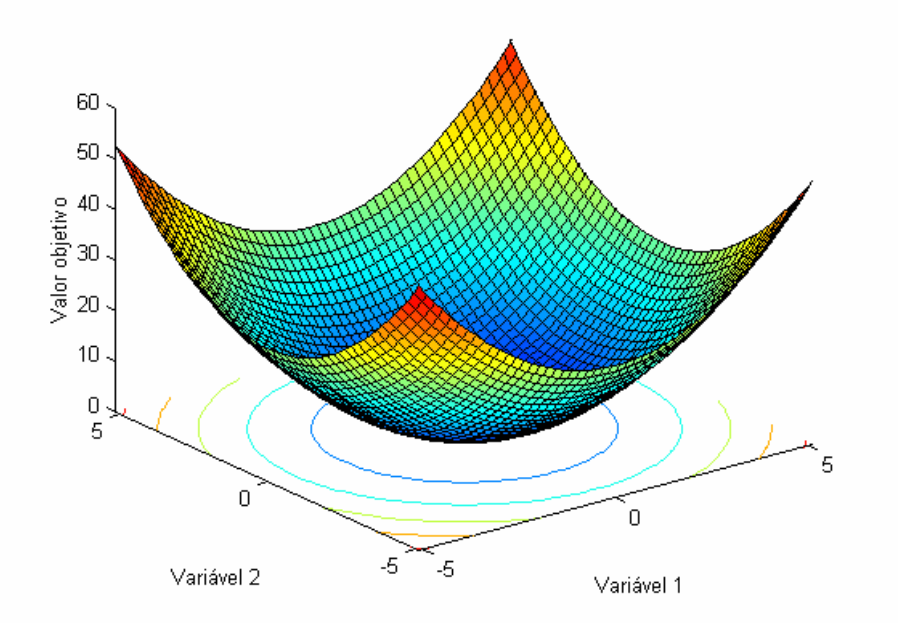

Figura 72. Visualização da função DeJong F1.

8.1.1 Otimização por Enxame de Partículas (PSO)

O algoritmo de otimização por Enxame de Partículas (PSO) foi utilizado para buscar o mínimo da função DeJong F1. Propõe-se que os indivíduos obtidos durante as gerações do processo de otimização sejam utilizados como exemplos de treinamento para uma rede neural a fim de que esta torne-se capaz de aproximar os resultados da função estudada.

Para que seja possível treinar a rede para aproximar a função corretamente é necessário gerar exemplos suficientes de forma que o espaço onde a função está definida esteja bem representado. Sendo assim, precisa-se verificar o número de indivíduos necessário na população do processo de otimização de forma que a rede possa ser treinada, com o menor número de exemplos de treinamento.

Para efeito de comparação dos exemplos gerados foram feitas três execuções diferentes do algoritmo de otimização, variando o número de indivíduos na população do PSO, com a mesma semente aleatória, ou seja, os indivíduos em comum da população inicial entre as três execuções foram sempre os mesmos.

Utilizou-se o algoritmo PSO com variação não-linear da inércia dada pela equação (9) com os parâmetros $\omega_0 = 3$ e $n = 1{,}2$. Foram utilizados somente os coeficientes de agregação C_1 e de congregação C_3 variando linearmente de acordo com as equações (10) e (12), respectivamente, onde $C_{1_{ini}} = 1$, $C_{1_{fin}} = 2$, $C_{3_{ini}} = 0$ e $C_{3_{fin}} = 1$.

Os números de indivíduos na população do processo de otimização testados foram 10, 20 e 30. Em todos os casos, foi considerado como critério de parada um número máximo de 200 gerações.

O mínimo global da função DeJong F1 é: $F_1(x_1, x_2, x_3) = 0, \quad x_i = 0, \quad 1 \le i \le 3$.

Os valores "ótimos" encontrados em cada processo de otimização por Enxame de Partículas testado podem ser vistos na Tabela 18. A função de *fitness* (objetivo) deste problema é a própria função DeJong F1, cujo valor pretende-se minimizar.

Tabela 18. Resultados do PSO na busca do valor mínimo da função DeJong F1.

Indivíduos	x_1	x_2	x_3	***Fitness***
10	-1,832e-04	3,288e-04	3,931e-04	2,962e-07
20	1,830e-04	8,466e-05	1,467e-04	6,216e-08
30	-2,351e-03	4,036e-04	-1,639e-04	5,717e-06

Na Tabela 18 pode-se ver que a otimização com 20 indivíduos na população chegou ao melhor resultado, obtendo o menor valor de *fitness* dos casos testados. Além disso, como o ponto de parada definido para o processo de otimização foi o número de gerações, a execução do algoritmo com 20 indivíduos realizou, no máximo, 4000 avaliações da função objetivo enquanto as execuções com 10 e 30 indivíduos realizaram cerca de 2000 e 6000 avaliações, respectivamente. A Figura 73 mostra a evolução de cada execução da otimização testada no detalhe entre 500 e 2000 avaliações da função objetivo. Pode-se ver que, inicialmente, a convergência do processo com 10 indivíduos é mais lenta devido à menor diversidade de alternativas para o melhor global da geração. No entanto, quando um novo melhor global é encontrado, a convergência dos outros indivíduos para este ponto é muito mais rápida.

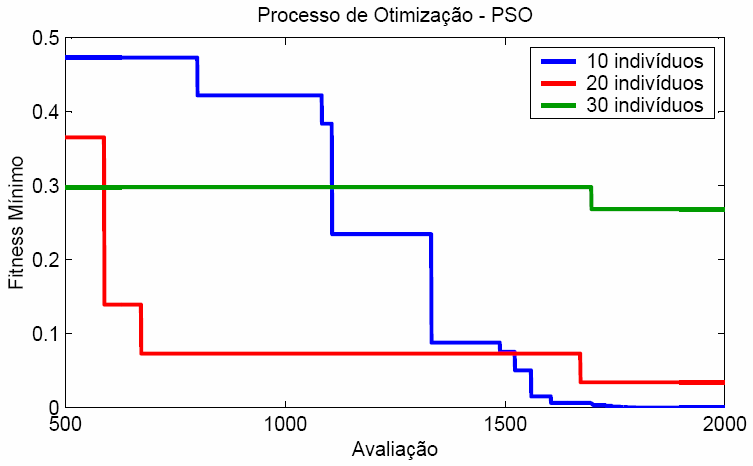

Figura 73. (Avaliação x *Fitness*): Comparação dos casos testados.

A seguir será analisado se o conjunto de indivíduos obtidos durante o processo de otimização pode fornecer informação suficiente para o treinamento de uma rede neural de forma que esta torne-se capaz de aproximar corretamente a função DeJong F1.

8.1.2 Treinamento da Rede Neural

Para a aproximação da função DeJong F1 foram utilizadas redes neurais *feedforward multilayer perceptron* com três tipos de configurações de rede com uma camada intermediária: com 4, 8 e 12 neurônios. Foram utilizadas a função de transferência tangente hiperbólica na camada intermediária e a função de transferência linear na camada da saída da rede.

Cada conjunto de dados obtido através da otimização apresentada na seção anterior foi dividido em três conjuntos: treinamento, validação e teste. Todos os dados de entrada e saída foram normalizados dentro do intervalo [-1,1].

Para efeito de comparação foram feitos dois tipos de análise. O primeiro tipo visou a escolha da melhor configuração para a rede neural (com 4, 8 ou 12 neurônios na camada intermediária) para cada um dos três conjuntos de dados avaliado.

No segundo tipo de análise, foi feita a comparação das melhores redes obtidas para cada um dos conjuntos de dados levando-se em consideração o erro médio quadrático das aproximações destas redes para valores que abrangem o espaço onde a função foi definida. Tais valores podem não fazer parte, necessariamente, dos conjuntos de dados obtidos na otimização.

Ao todo foram testados 27 casos, sendo três para cada topologia de rede para cada um dos três conjuntos de dados testado. A Tabela 19 mostra as especificações de cada um dos 27 casos estudados. Em todos os casos foram utilizados conjuntos de validação com 20 exemplos para evitar *overfitting* na rede.

O treinamento das redes foi realizado com o algoritmo *Levenberg-Marquardt backpropagation* (HAGAN & MENHAJ, 1994). Considerou-se um limite máximo de 200 épocas, visando um erro médio quadrático mímino no conjunto de treinamento de 10^{-5}. Além disso, se o erro no conjunto de validação aumentar por 6 épocas seguidas, o treinamento é concluído. Cada configuração de rede foi treinada em 20 execuções independentes, com pesos iniciais aleatórios diferentes.

Tabela 19. Casos testados para a aproximação da função DeJong F1.

Indivíduos na População	Número de Avaliações	Exemplos de Treinamento	Exemplos de Teste	Neurônios	Caso
10	2000	50	1930	4	1
				8	2
				12	3
		80	1900	4	4
				8	5
				12	6
		120	1860	4	7
				8	8
				12	9
20	4000	50	3930	4	10
				8	11
				12	12
		80	3900	4	13
				8	14
				12	15
		120	3860	4	16
				8	17
				12	18
30	6000	50	5930	4	19
				8	20
				12	21
		80	5900	4	22
				8	23
				12	24
		120	5860	4	25
				8	26
				12	27

Na Tabela 20 são apresentados os resultados médios de erro e desvio padrão nos conjuntos de treinamento, validação e teste, assim como o número de épocas e tempo médio de treinamento considerando as 20 execuções do algoritmo para cada caso estudado. Pode-se ver que, dentre todos os casos analisados, o caso 9 obteve o menor erro médio, além do menor desvio padrão médio, em todos os conjuntos (treinamento, validação e teste). Entretanto, no caso 9 o conjunto de testes é composto por 1860 exemplos, e nos casos 18, 24 e 27, que também obtiveram erros na ordem de 10^{-5}, os conjuntos de testes são compostos por 3860, 5900 e 5860 exemplos, respectivamente.

É possível observar que a rede composta por 4 neurônios na camada intermediária não é suficiente para aproximar a função, tendo obtido os maiores valores de erro em todos os casos, tanto para o conjunto de teste quanto para os conjuntos de treinamento e validação, o que sugere que o algoritmo parou muito antes de atingir o valor mínimo de

erro desejado, provavelmente devido ao aumento do erro no conjunto de validação (critério de parada). Nas redes utilizando 8 neurônios na camada intermediária os resultados foram significativamente melhores que os encontrados utilizando 4 neurônios. No entanto, os melhores resultados, para todos os conjuntos de dados testados, foram obtidos utilizando 12 neurônios na camada intermediária. A Figura 74 mostra os gráficos com os históricos de treinamento de redes neurais com 4, 8 e 12 neurônios na camada intermediária, utilizando os dados obtidos na otimização com 30 indivíduos na população e 120 exemplos de treinamento (casos 25, 26 e 27, respectivamente).

Tabela 20. Resultados do treinamento das redes neurais em cada caso estudado.

Caso	Erro de Treinamento		Erro de Validação		Erro de Teste		Épocas	Tempo (s)
	μ	σ	μ	σ	μ	σ		
1	1,834e-02	2,624e-02	3,048e-02	2,732e-02	3,636e-02	4,971e-02	25	0,642
2	4,820e-04	1,714e-03	3,203e-03	9,537e-03	1,825e-03	4,351e-03	50	1,309
3	6,636e-04	2,697e-03	1,830e-03	3,986e-03	1,900e-03	4,542e-03	27	1,131
4	4,413e-02	7,724e-02	7,360e-02	6,525e-02	8,492e-02	1,460e-01	31	1,463
5	1,222e-04	2,706e-04	1,169e-03	1,682e-03	4,776e-04	7,752e-04	64	2,712
6	9,211e-05	1,986e-04	2,280e-03	4,615e-03	4,039e-04	6,464e-04	31	1,747
7	2,474e-02	4,221e-02	2,333e-02	5,007e-02	3,606e-02	7,383e-02	55	2,252
8	2,733e-04	9,012e-04	2,020e-04	6,154e-04	3,224e-04	8,135e-04	69	2,959
9	2,389e-05	3,080e-05	3,077e-05	2,795e-05	**6,860e-05**	5,522e-05	52	2,684
10	1,779e-02	1,930e-02	2,944e-02	2,725e-02	4,255e-02	4,257e-02	50	1,580
11	4,101e-05	7,762e-05	3,747e-04	4,169e-04	6,953e-04	8,350e-04	67	2,578
12	3,115e-05	4,207e-05	3,684e-04	5,012e-04	7,943e-04	1,269e-03	27	1,683
13	2,365e-02	2,694e-02	2,913e-02	3,328e-02	4,394e-02	5,450e-02	57	2,645
14	1,219e-04	2,475e-04	2,919e-04	4,514e-04	2,748e-04	4,168e-04	74	3,973
15	3,377e-05	7,329e-05	1,018e-04	1,206e-04	1,173e-04	1,604e-04	49	3,294
16	2,078e-02	2,144e-02	2,280e-02	2,093e-02	3,262e-02	3,550e-02	62	3,071
17	2,395e-03	1,019e-02	2,127e-03	7,972e-03	4,448e-03	1,893e-02	79	4,449
18	3,930e-05	5,300e-05	1,597e-04	8,187e-05	**9,977e-05**	1,143e-04	45	3,218
19	2,731e-02	2,537e-02	3,600e-02	3,271e-02	6,681e-02	6,555e-02	43	1,952
20	2,528e-05	3,340e-05	1,481e-04	1,714e-04	2,162e-04	2,014e-04	72	3,887
21	3,371e-05	9,391e-05	2,128e-04	2,390e-04	2,939e-04	5,078e-04	40	3,160
22	3,067e-02	2,211e-02	3,501e-02	2,448e-02	4,457e-02	3,481e-02	49	2,806
23	7,040e-04	2,967e-03	6,494e-04	2,213e-03	1,281e-03	5,164e-03	80	5,184
24	2,987e-05	3,661e-05	6,631e-05	6,937e-05	**8,708e-05**	1,070e-04	52	4,334
25	3,413e-02	3,276e-02	5,230e-02	4,451e-02	4,550e-02	5,824e-02	46	2,697
26	2,400e-04	7,239e-04	2,458e-04	4,821e-04	5,014e-04	1,567e-03	87	5,474
27	3,764e-05	6,232e-05	8,627e-05	1,430e-04	**7,062e-05**	1,033e-04	64	5,262

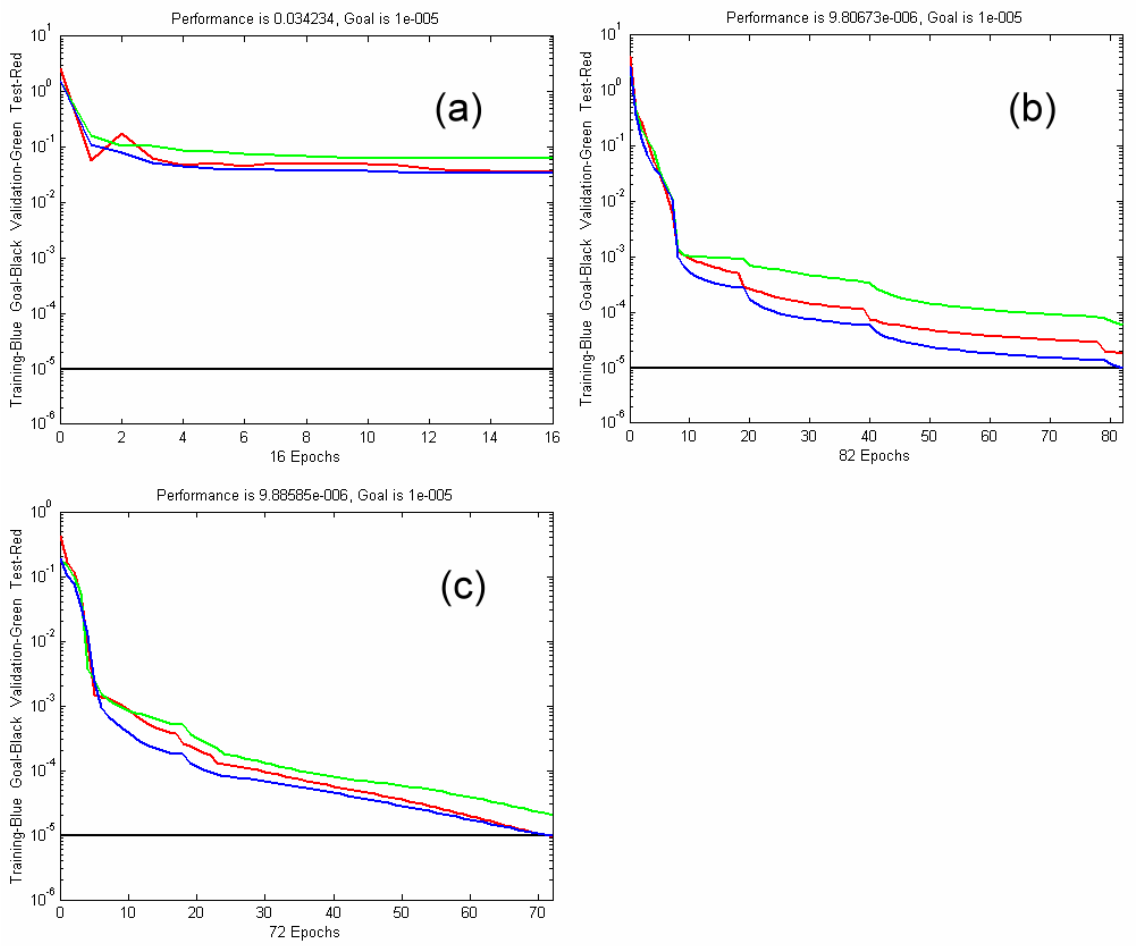

Figura 74. Históricos de treinamento: (a) caso 25; (b) caso 26; (c) caso 27.

A Figura 75 mostra os gráficos com os históricos de treinamento de redes neurais com 12 neurônios na camada intermediária, utilizando os dados obtidos na otimização com 10 indivíduos na população. A Figura 75a considera um conjunto de treinamento com 50 exemplos e a Figura 75b, um conjunto de treinamento com 120 exemplos. Pode-se observar que, com relação ao tamanho do conjunto de treinamento, aquele com 50 exemplos não possui informação suficiente para treinar a rede apropriadamente, resultando em erros maiores no conjunto de teste do que os obtidos utilizando 120 exemplos no treinamento.

Considerando os casos que utilizaram o conjunto de exemplos gerado na otimização com 20 indivíduos na população, que atingiu o menor valor objetivo (*fitness*) dos três conjuntos de teste estudados, o caso 18 foi o que aproximou melhor os valores do conjunto de teste. Entre os casos que utilizaram os dados com 10 indivíduos na população, o caso 9 obteve menor erro médio no conjunto de teste. Já, nos casos utilizando 30 indivíduos na população, a diferença entre o erro médio no conjunto de teste do caso 27 para o caso 24 não foi estatisticamente relevante. Assim, precisa-se

comparar esses dois casos com relação a um outro parâmetro a fim de saber qual deles alcançou a melhor aproximação da função DeJong F1.

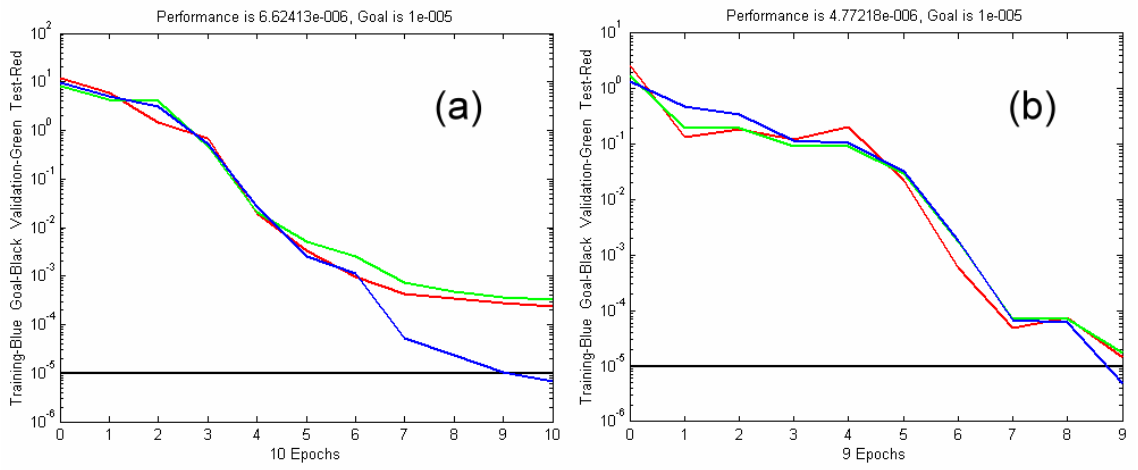

Figura 75. Históricos de treinamento: (a) caso 7; (b) caso 9.

A Figura 76 mostra uma comparação entre os valores reais (Y_{exato}) e os valores aproximados pela redes neurais (Y_{RN}) dos casos 24 e 27, respectivamente. Utilizou-se como base de comparação a rede neural que obteve o melhor resultado, entre as 20 execuções de treinamento, para cada caso. Pode-se ver claramente que a função é melhor aproximada no caso 27.

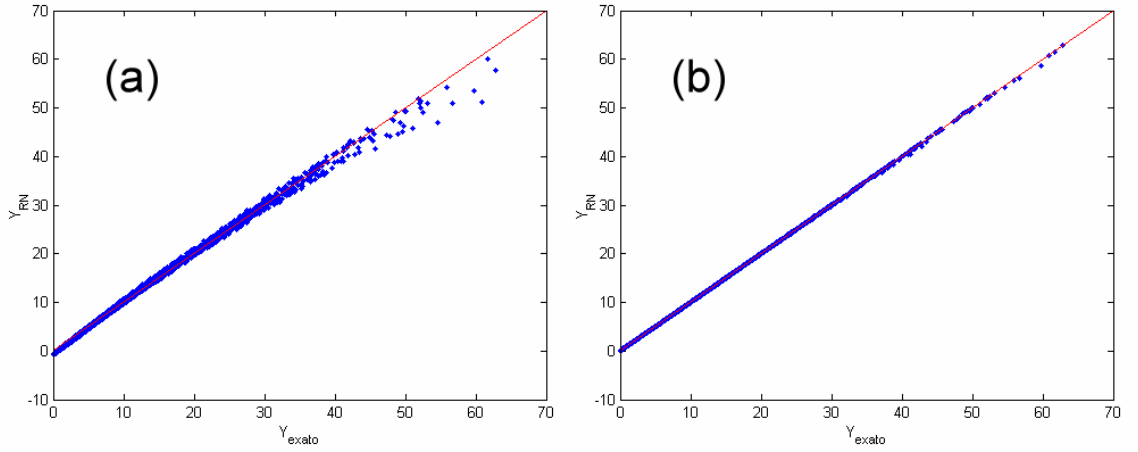

Figura 76. (Y_{exato} x Y_{RN}): (a) caso 24 e (b) caso 27.

Então, conclui-se que os casos 9, 18 e 27 obtiveram os melhores resultados de aproximação da função DeJong F1 considerando os conjuntos de dados gerados com 10, 20 e 30 indivíduos na população da otimização, respectivamente.

Agora, para analisar qual das três redes neurais treinadas escolhidas aproxima melhor a função (segunda análise), precisa-se verificar o erro encontrado quando as redes são utilizadas para aproximar vários pontos que abrangem o espaço onde a função foi definida. Para isso, foram gerados 200 novos pontos da seguinte maneira: 50 na seção ($x(i)$,0,0), 50 na seção (0,$x(i)$,0), 50 na seção (0,0,$x(i)$) e 50 na seção ($x(i)$,$x(i)$,$x(i)$), onde i =1, 2,..., 50 e x é um vetor de pontos equidistantes dado por:

x = [-5,1200 -4,9110 -4,7020 -4,4931 -4,2841 -4,0751 -3,8661 -3,6571 -3,4482 -3,2392 -3,0302 -2,8212 -2,6122 -2,4033 -2,1943 -1,9853 -1,7763 -1,5673 -1,3584 -1,1494 -0,9404 -0,7314 -0,5224 -0,3135 -0,1045 0,1045 0,3135 0,5224 0,7314 0,9404 1,1494 1,3584 1,5673 1,7763 1,9853 2,1943 2,4033 2,6122 2,8212 3,0302 3,2392 3,4482 3,6571 3,8661 4,0751 4,2841 4,4931 4,7020 4,9110 5,1200];

Assim, as redes neurais que obtiveram os menores erros no conjunto de teste nos casos 9, 18 e 27 foram aplicadas na aproximação destes 200 novos exemplos. Os erros obtidos nestes novos testes podem ser vistos na Tabela 21.

Tabela 21. Resultados das redes na aproximação da função DeJong F1.

Caso	Erro Médio
9	2,800e-03
18	1,869e-04
27	5,499e-06

A Tabela 21 mostra que a rede treinada com os dados da otimização com 30 indivíduos na população e com 12 neurônios na camada intermediária é a melhor opção para aproximar a função DeJong F1. Isso era esperado uma vez que, tendo mais indivíduos em cada geração da otimização, provavelmente, os exemplos gerados irão abranger um espaço de solução maior. O caso 9, que havia obtido o menor erro médio na análise anterior, foi o que apresentou o pior resultado na aproximação da função.

As Figuras 77 e 78 mostram as aproximações obtidas pelas redes neurais nos casos 9 e 27, respectivamente, na seção 2D (x,x,x) da função DeJong F1. As imagens dos gráficos nas outras seções 2D testadas foram suprimidas porque, nessas seções, não havia uma diferença visualmente relevante entre o valor da função e as aproximações dadas pela rede em ambos os casos. As Figuras 79 e 80 mostram as superfícies de erro nos casos 9 e 27, respectivamente. Pode-se ver que o erro absoluto máximo do caso 27 é de cerca de 0,8 e, no caso 9, mais de 3.

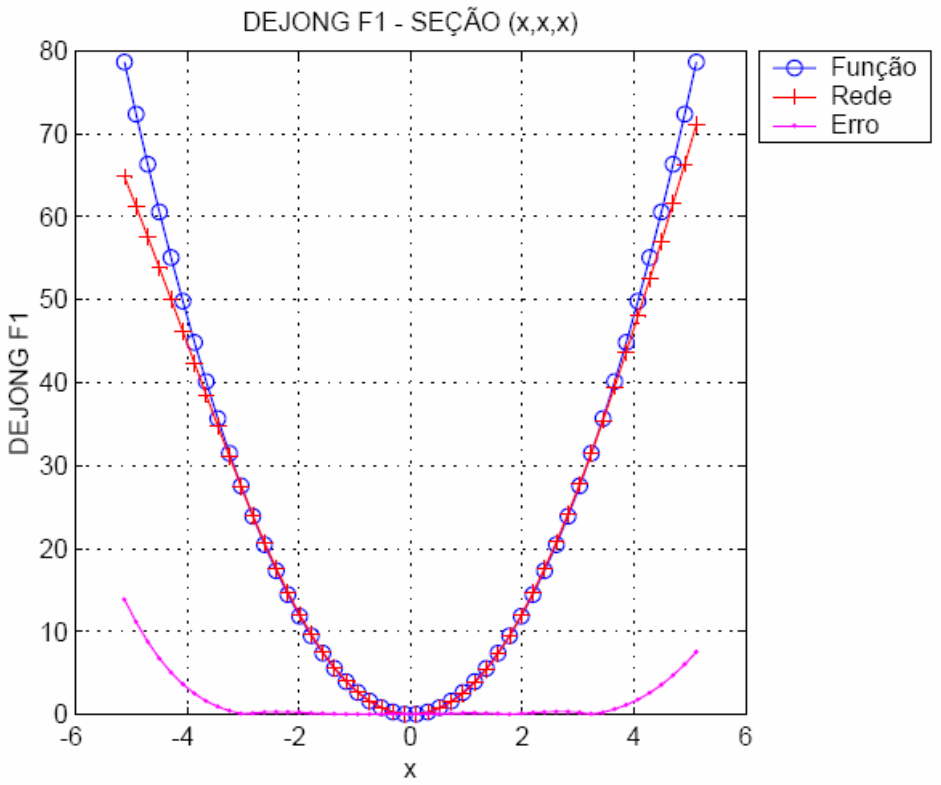

Figura 77. Aproximação da função DeJong F1 no caso 9 – Seção 2D (*x*,*x*,*x*).

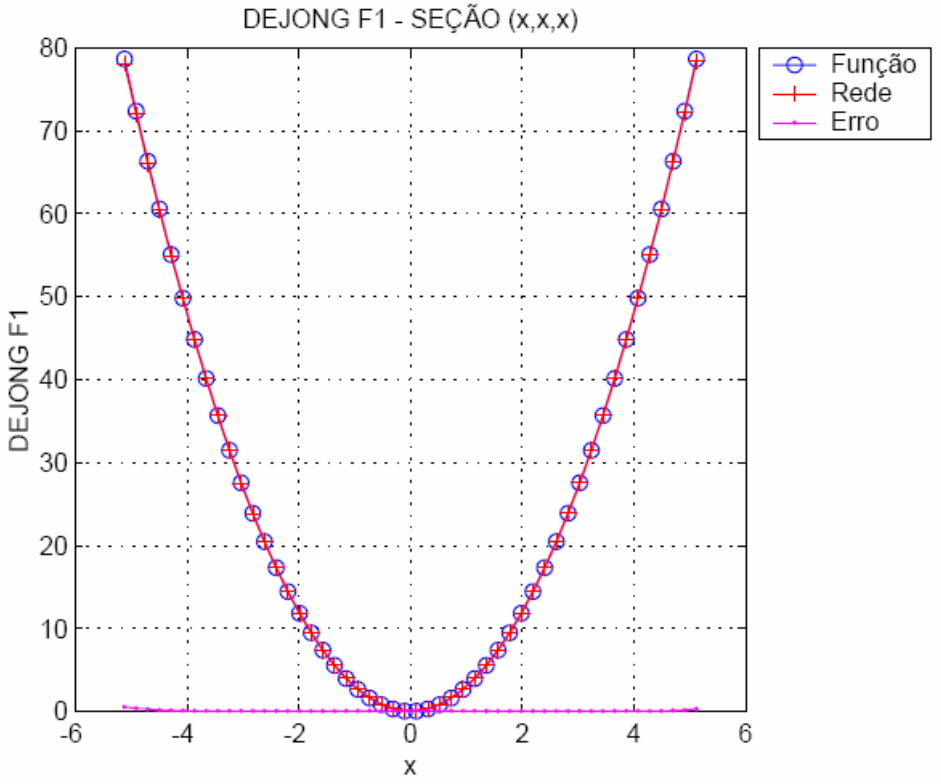

Figura 78. Aproximação da função DeJong F1 no caso 27 – Seção 2D (*x*,*x*,*x*).

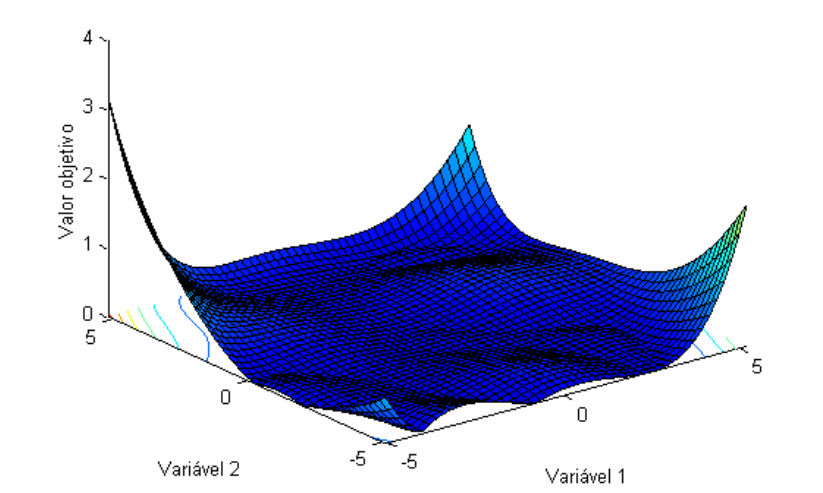

Figura 79. Gráfico do erro de aproximação da função DeJong F1 no caso 9.

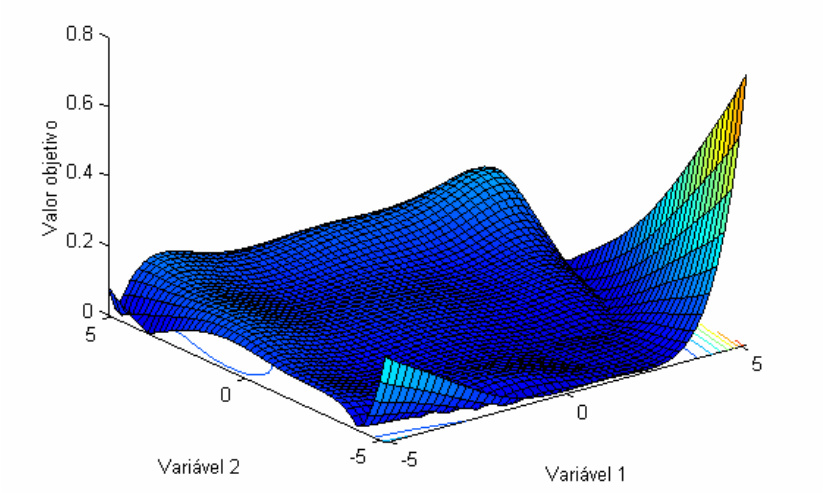

Figura 80. Gráfico do erro de aproximação da função DeJong F1 no caso 27.

8.2 Sistemas de Ancoragem

Utilizou-se o programa SITUA/PROSIM para gerar e analisar o modelo de sistemas de ancoragem apresentado neste trabalho. Foi utilizado um modelo, denominado P1, para representar o sistema de ancoragem de uma plataforma hipotética com 13 linhas, sendo 8 linhas de ancoragem e 5 *risers*. O mesmo modelo foi utilizado por ALBRECHT (2005) e MONTEIRO (2008).

Tomando por base a estratégia de metamodelagem com aproximação sequencial, onde um metamodelo global é ajustado e então usado como um substituto da função custosa, o objetivo do estudo aqui apresentado é verificar a possibilidade de se substituir a análise de uma configuração de sistema de ancoragem, em geral realizada por meio de análise estática ou dinâmica, por um metamodelo de rede neural que aproxime

diretamente a resposta da análise rigorosa (ou seja, os valores de saída da otimização). Essa rede neural utiliza como parâmetros de entrada e saída da rede os mesmos parâmetros do algoritmo de otimização. Sendo assim, para obter exemplos para o treinamento da rede neural, foi aplicado o algoritmo de PSO na otimização da configuração do sistema de ancoragem estudado. As configurações candidatas obtidas durante o processo de otimização foram utilizadas para treinamento, validação e teste das redes neurais artificiais.

Como se trata de analisar a viabilidade da utilização da estratégia de metamodelagem com aproximação sequencial na otimização de projeto de um sistema de ancoragem, foi utilizada somente a análise estática para a descrição do movimento da embarcação. Com o uso da análise estática as avaliações são mais rápidas.

8.2.1 Descrição do Modelo

O modelo P1 é simples, porém assimétrico, composto por 8 linhas de ancoragem, com arranjo inicial simétrico e 5 *risers* com arranjo assimétrico, todos voltados para Norte.

Os dados do casco utilizado neste modelo, mostrado na Figura 81, são:

- Peso: 23800 toneladas;
- Calado de projeto: 22,2 m;
- Comprimento: 106 m
- Boca: 65 m;

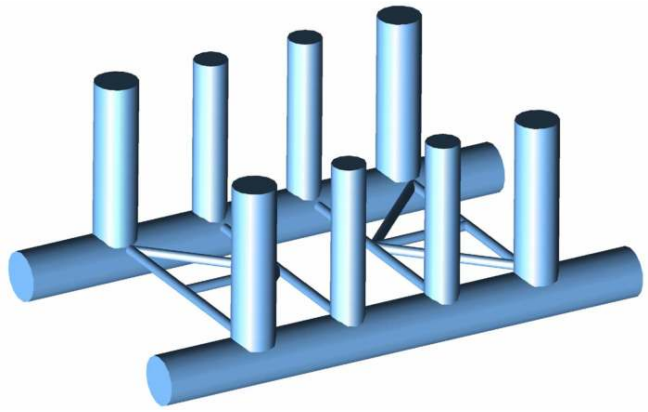

Figura 81. Casco da plataforma P1.

A lâmina d'água utilizada foi de 2400m e as linhas de ancoragem foram divididas em três segmentos utilizando amarra nas extremidades e cabo de poliéster no trecho intermediário. A composição das linhas pode ser vista na Tabela 22.

Tabela 22. Composição das linhas do modelo P1

Segmento	Comprimento (m)	Material	Diam (m)	MBL (kN)
Topo	250	Amarra	0,16	21234,00
Intermediário	3000	Poliéster	0,20	10987,00
Fundo	700	Amarra	0,16	21234,00

Para os *risers*, foi escolhido um material de *riser* flexível, com 0,277m de diâmetro do banco de dados do SITUA.

A Figura 82a mostra o arranjo das linhas no modelo P1. A Figura 82b mostra todo o sistema de forma tridimensional, onde pode-se perceber claramente a assimetria do sistema.

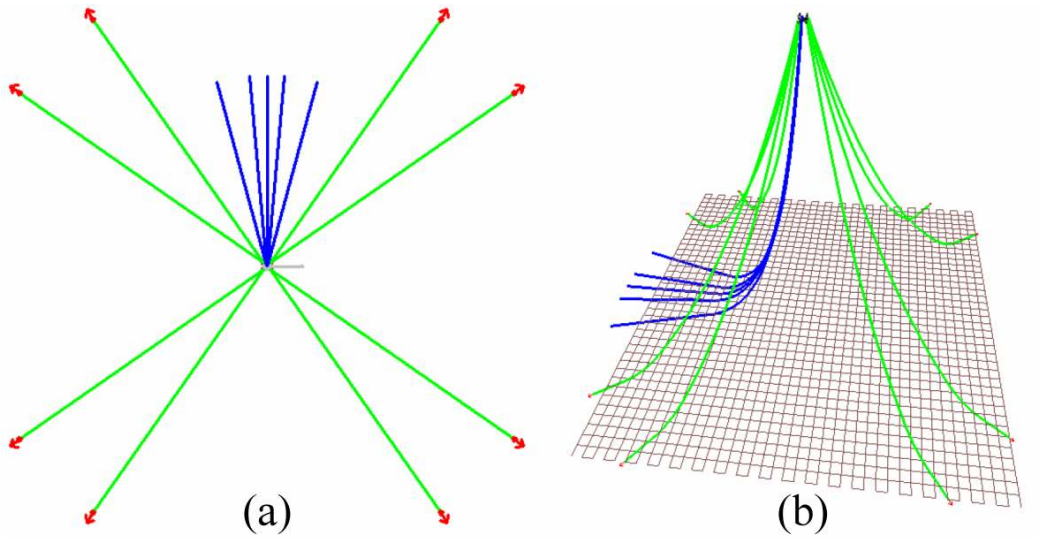

Figura 82. Modelo P1: (a) Arranjo das linhas; (b) Vista tridimensional.

8.2.2 Dados ambientais

Serão aplicados os carregamentos ambientais de correnteza e a parcela estática do vento, uma vez que será efetuada apenas a análise estática.

Os carregamentos de correnteza foram aplicados em oito direções: Norte, Nordeste, Leste, Sudeste, Sul, Sudoeste, Oeste e Noroeste, com período de recorrência de 100 anos.

Para o carregamento de vento foi utilizado o período de recorrência de 10 anos.

Tais carregamentos foram obtidos pelo SITUA através da importação de dados Meteo-Oceanográficos da ET-3000.00-1000-941-PPC-001 (PETROBRAS,1999).

O fundo utilizado no modelo considerado é plano, com inclinação de 0º e numa profundidade de 2400m.

8.2.3 Definição do Problema de Otimização de Sistemas de Ancoragem

Variáveis de Projeto

Um modelo de ancoragem possui diversas variáveis que podem funcionar como variáveis de projeto num problema de otimização. Para simplificar o problema foram escolhidas três variáveis: o raio de ancoragem, a tração de trabalho média e o azimute das linhas.

Neste trabalho, a composição de cada linha, em termos do tipo de material utilizado e número de segmentos, permaneceu inalterada.

O modelo foi considerado com agrupamento das linhas de ancoragem por corner (MONTEIRO, 2008). Dessa forma, o modelo possui 9 dimensões: o azimute dos 4 corners, o raio de ancoragem para cada corner e a tração de trabalho média.

Função Objetivo e Restrição

A principal finalidade do sistema de ancoragem é manter a posição da plataforma. Assim, a função objetivo a ser utilizada é o próprio passeio (*offset*) da embarcação, que deseja-se que seja mínimo. Dessa forma, utilizou-se a seguinte expressão para a função objetivo:

$$f = e^{k*\frac{offset}{profund}} \tag{70}$$

onde *offset* é o maior *offset*, em metros, da embarcação dentre as 8 direções analisadas; *profund* é a profundidade, em metros, da locação analisada; e *k* é o fator de ajuste.

Utilizou-se k = 8, pois, segundo MONTEIRO (2008), com *offsets* pequenos a variação da função mostra-se mais significativa com este valor. Isso pode ser observado na Figura 83.

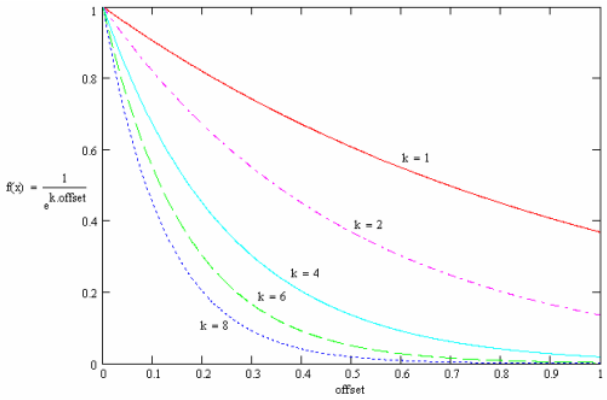

Figura 83. Escolha do valor de *k* (MONTEIRO, 2008).

O aumento da tração de trabalho ou a diminuição do raio de ancoragem levará a uma diminuição do *offset*, no entanto existem limites para a tração em cada linha. A API (2001) recomenda que a tração a que a linha está submetida não ultrapasse 50% da tração de ruptura da linha (*Maximum Breaking Load* - MBL) para análises estáticas. Além disso, a tração não deve ser inferior 30% da tração de ruptura da linha (MBL) (ALBRECHT, 2005).

Portanto, a função objetivo penalizada deve levar em consideração o *offset* e a relação entre a tração atuante na linha e a tração máxima admitida. Sendo assim, utilizou-se como funções de penalidade as seguintes expressões:

$$Pen03 = \begin{cases} e^{0,3-\frac{TracMax}{MBL}}, \ se \quad \dfrac{TracMax}{MBL} \leq 0,3 \\ 0, \ se \quad \dfrac{TracMax}{MBL} > 0,3 \end{cases} \tag{71}$$

e

$$Pen06 = \begin{cases} e^{\frac{TracMax}{MBL}-0,5}, \ se \quad \dfrac{TracMax}{MBL} \geq 0,5 \\ 0, \ se \quad \dfrac{TracMax}{MBL} < 0,5 \end{cases} \tag{72}$$

onde *TracMax* é a tração da linha mais solicitada dentre todas as linhas e todas as direções analisadas; e *MBL* é a tração máxima de ruptura da linha (*Maximum Breaking Load*).

A função objetivo penalizada (função de *fitness*) é descrita por:

$$f_{obj} = \frac{1}{f + Pen03 + Pen06} \tag{73}$$

Assim, quanto maior for o valor da função objetivo, menor será o *offset* da embarcação.

8.2.4 Otimização por Enxame de Partículas

Utilizou-se o programa *ProgOtim* para a otimização dos sistemas de ancoragem. O *ProgOtim* lê os modelos gerados no SITUA e, a partir deles, são gerados os indivíduos utilizando as variáveis de projeto citadas na seção anterior que são o azimute dos 4 corners, o raio de ancoragem para cada corner e a tração de trabalho média. Sendo assim, serão ao todo 9 parâmetros de entrada (variáveis de projeto), onde os valores de azimute e raio são dados em função da variação permitida para esses parâmetros a partir

da configuração inicial do sistema. A Tabela 23 mostra a variação possível para estas variáveis. O ângulo alfa refere-se à variação permitida para o azimute das linhas.

Tabela 23. Valores extremos das variáveis de projeto.

Variável	Mínimo	Máximo
Ângulo Alfa (^{o})	-5	5
Variação do Raio (m)	-200	200
Tração (kN)	1000	6000

O algoritmo de otimização por Enxame de Partículas (PSO) foi aplicado no problema de projeto de sistemas de ancoragem na busca por uma configuração "ótima" para esses sistemas, ou seja, a fim de maximizar o valor da função de *fitness* dada pela equação (73).

Na análise utilizada, para cada configuração candidata gerada no processo de otimização são executadas oito análises acopladas onde a maior tração no topo e o maior *offset* obtidos são usados para calcular a *fitness*. Esses três parâmetros, portanto, constituem a resposta do sistema e o valor de *fitness* é utilizado para classificar o indivíduo na população. O valor de *offset* é dado em percentagem da lâmina d'água e o valor de tração, em percentagem da MBL. Para poupar custos computacionais, tendo em vista que a análise dinâmica é muito onerosa, utilizou-se apenas análise estática.

Assim como no caso da aproximação de funções matemáticas apresentado anteriormente, pretende-se criar um conjunto de dados utilizando os indivíduos (configurações canditatas) obtidos durante as gerações do processo de otimização para ser utilizado como conjunto de treinamento para uma rede neural. Para que seja possível treinar uma rede a fim de ser um metamodelo global para substituir a análise do sistema de ancoragem, precisa-se gerar exemplos de forma que o espaço de busca do problema seja suficientemente explorado. Dessa forma, torna-se necessário verificar o número de indivíduos necessário na população do processo de otimização de forma que a rede possa ser treinada adequadamente, utilizando o menor número de exemplos de treinamento possível.

A fim de definir o melhor tamanho para a população da otimização, foram executadas 5 otimizações completas variando o número de indivíduos na população inicial do PSO e utilizando análise estática na avaliação das configurações candidatas. Utilizou-se, em todas as 5 execuções, uma semente aleatória igual, ou seja, os indivíduos em comum da população inicial entre as cinco execuções, foram sempre os

mesmos. Os números de indivíduos testados na população do processo de otimização foram 10, 20, 30, 40 e 50. Na fase de treinamento das redes neurais, todas as configurações testadas durante os processos de otimização serão divididas em dados de treinamento, validação e teste.

Utilizou-se o algoritmo PSO com variação não-linear da inércia dada pela equação (9) com os parâmetros $\omega_0 = 3$ e $n = 1{,}2$. Foram utilizados somente os coeficientes de agregação C_1 e de congregação C_3 variando linearmente de acordo com as equações (10) e (12), respectivamente, onde $C_{1_{ini}} = 1$, $C_{1_{fin}} = 2$, $C_{3_{ini}} = 0$ e $C_{3_{fin}} = 1$. Não foi realizada uma análise paramétrica do PSO para otimização de sistemas de ancoragem. No entanto, testes preliminares mostraram que os valores obtidos para os parâmetros na customização dos *risers* apresentada no capítulo 6, não retornam os melhores resultados para sistemas de ancoragem. Sendo assim, optou-se por utilizar os valores definidos anteriormente.

A execução do algoritmo é concluída quando o valor médio de *fitness* permanece acima de 95% do melhor valor de *fitness* encontrado ao longo de três iterações consecutivas, significando que as partículas estão concentradas perto de uma solução ótima. Caso não haja esse tipo de convergência, o processo é encerrado ao atingir um número máximo de 30 gerações.

O modelo utilizado, antes de ser otimizado, apresenta um *offset* máximo de 319m, que representa 13,3% da lâmina d'água, e a linha mais tracionada está com 37,5% da MBL.

As soluções "ótimas" encontradas em cada processo de otimização por Enxame de Partículas testado podem ser vistas na Tabela 24.

Tabela 24. Resultados do PSO na otimização de sistemas de ancoragem.

Indivíduos	*Offset* Máx. (%)	Tração Máx. (%)	*Fitness*	N. Avaliações
10	3,57	40,01	0,7517	120
20	3,49	41,16	**0,7567**	280
30	3,54	40,89	0,7534	360
40	3,36	41,17	**0,7645**	520
50	3,36	39,16	0,7642	800

Pode-se observar na Tabela 24 que, considerando somente a eficiência do algoritmo (em termos da *fitness*), o caso utilizando 40 indivíduos na população da otimização obteve o melhor resultado, representando um ganho de 74% no *offset* da

plataforma e admitindo uma tração no topo máxima 9% maior que no sistema original. No entanto, tal resultado foi obtido através de 520 avaliações da função de *fitness*. Sendo assim, quando o desempenho computacional é considerado, o caso utilizando 20 indivíduos na população pode ser considerado o melhor, uma vez que realizou apenas 280 avaliações, atingindo um ganho maior do que no caso com 30 indivíduos.

A Tabela 25 mostra uma comparação entre a configuração original do modelo e a melhor configuração encontrada entre os processos de otimização executados (utilizando 40 indivíduos na população do PSO).

Tabela 25. Comparação entre o modelo original e o otimizado.

Linha	Original		Otimizado	
	Raio (m)	Azimute (^{o})	Raio (m)	Azimute (^{o})
1	3000,00	305,00	3128,45	304,00
2	3000,00	325,00	3128,45	324,00
3	3000,00	35,00	2800,00	38,85
4	3000,00	55,00	2800,00	58,85
5	3000,00	125,00	3200,00	120,09
6	3000,00	145,00	3200,00	140,09
7	3000,00	215,00	3101,76	220,00
8	3000,00	235,00	3101,76	240,00

8.2.5 Treinamento da Rede Neural

Todas as configurações testadas durante cada um dos 5 processos de otimização descritos na seção anterior foram divididas em dados de treinamento, validação e teste para serem utilizados no treinamento e análise de desempenho de redes neurais artificiais, visando a aproximação dos resultados da análise estática de sistemas de ancoragem.

Uma vez que o problema de otimização de sistemas de ancoragem retorna três valores de resposta que definem a qualidade da configuração (*offset* máximo, tração no topo máxima e *fitness*), propõe-se o estudo de três modelos de análise desse problema via redes neurais. A diferença básica entre esses modelos é a arquitetura da rede ultilizada com relação ao número de saídas.

O primeiro modelo proposto baseia-se na idéia de utilizar a rede neural simplesmente para substituir a análise estática, ou seja, para aproximar os valores de *offset* máximo e tração no topo máxima. Sendo assim, a rede utilizada nesse método terá duas saídas. Uma vez que cada indivíduo na população do PSO é classificado de acordo

com o seu valor de *fitness*, nesse método é necessário utilizar as equações (70) a (73) a fim de se obter esse valor. Esse modelo será citado como *TO* (*Tração e Offset*).

O segundo modelo baseia-se na idéia de utilizar a rede neural para substituir todos os cálculos envolvidos na aquisição dos parâmetros de resposta do sistema, ou seja, para aproximar diretamente os valores de *offset* máximo, tração no topo máxima e *fitness* obtidos no processo de otimização. Dessa forma, a rede neural utilizada terá três saídas e não serão necessários cálculos posteriores para realizar a classificação dos indivíduos na população. Esse modelo será citado como *TOF* (*Tração, Offset e Fitness*).

Durante a otimização, o único parâmetro de resposta do sistema que interfere no processo de evolução dos indivíduos é o valor de *fitness*, que atua diretamente na atualização do melhor global e do melhor local de cada indivíduo, além de ser utilizado no cálculo do centro de massa da população (Equação (5)). Por esse motivo, a proposta do terceiro modelo é utilizar a rede neural para aproximar diretamente apenas o valor de *fitness* das configurações candidatas. Nesse caso, a rede neural utilizada terá apenas uma saída. No final de um processo de otimização utilizando uma rede neural desse tipo, será necessário realizar uma análise estática da configuração considerada "ótima" a fim de obter os valores de *offset* máximo e tração no topo máxima do sistema de ancoragem otimizado. Esse modelo será citado como *F* (*Fitness*).

Foram treinadas redes neurais utilizando dois tipos de configurações de rede com uma camada intermediária: com 8 e 12 neurônios. Utilizou-se a função de transferência tangente hiperbólica na camada intermediária e a função de transferência linear na camada da saída da rede.

Ao todo foram testados 30 casos em cada modelo, sendo três para cada topologia de rede para cada um dos cinco conjuntos de dados obtidos na seção anterior. A Tabela 26 mostra as especificações de cada um dos 30 casos estudados. No que segue, serão referidos como Indiv_10, Indiv_20, Indiv_30, Indiv_40 e Indiv_50 os conjuntos de dados construídos durante os processos de otimização do PSO com 10, 20, 30, 40 e 50 indivíduos na população, respectivamente.

Os conjuntos de dados foram divididos de forma a utilizar na fase de treinamento, no máximo, 40%, 50% ou 60% do total de dados. Os dados utilizados na fase de treinamento foram divididos em 90% para treinamento e 10% para validação da rede. O restante dos dados foram utilizados para teste a fim de verificar a qualidade da rede treinada.

Os dados foram divididos por geração do processo de otimização, ou seja, os conjuntos "Treinamento+Validação" e "Teste" são múltiplos do número de indivíduos na população que gerou os dados testados. Isso foi feito baseado na idéia de interromper a utilização da análise rigorosa, substituindo-a pela rede neural, ao fim de um certo número de gerações do processo de otimização.

Tabela 26. Casos testados para a otimização do sistema de ancoragem.

Conjunto de Dados	Número de Avaliações	Dados na Fase de Treino	Exemplos de Treinamento	Exemplos de Validação	Exemplos de Teste	Neurônios	Caso
Indiv_10	120	40%	36	4	80	8	1
						12	2
		50%	54	6	60	8	3
						12	4
		60%	63	7	50	8	5
						12	6
Indiv_20	280	40%	90	10	180	8	7
						12	8
		50%	126	14	140	8	9
						12	10
		60%	144	16	120	8	11
						12	12
Indiv_30	360	40%	108	12	240	8	13
						12	14
		50%	162	18	180	8	15
						12	16
		60%	189	21	150	8	17
						12	18
Indiv_40	520	40%	180	20	320	8	19
						12	20
		50%	216	24	280	8	21
						12	22
		60%	252	28	240	8	23
						12	24
Indiv_50	800	40%	270	30	500	8	25
						12	26
		50%	360	40	400	8	27
						12	28
		60%	405	45	350	8	29
						12	30

Em todos os testes realizados para cada modelo proposto, o treinamento das redes foi realizado com o algoritmo *Levenberg-Marquardt backpropagation*. Considerou-se um limite máximo de 200 épocas, visando um erro médio quadrático mímino no conjunto de treinamento de 10^{-5}. Além disso, se o erro no conjunto de validação aumentar por 6 épocas seguidas, o treinamento é concluído. Cada configuração de rede foi treinada em 20 execuções independentes, com pesos iniciais aleatórios diferentes.

A seguir serão apresentadas as análises de desempenho dos três modelos propostos.

Modelo 1 - Tração e Offset como saídas para a Rede Neural (TO)

Nesse primeiro modelo de análise, foram utilizadas redes neurais com nove entradas e duas saídas (*offset* e tração no topo máximos). Os resultados estatísticos das realizações dos experimentos são apresentados na Tabela 27, em termos de média (μ) e desvio padrão (σ) dos erros obtidos nos conjuntos de treinamento, validação e teste, assim como o número de épocas e tempo médio necessários para o treinamento. Os valores de erro obtidos correspondem ao erro médio quadrático de todos os valores de saída da rede.

Na Tabela 27 pode-se ver que, em todos os casos avaliados, os melhores resultados foram obtidos utilizando 60% dos dados na fase de treinamento. No entanto, a diferença entre os resultados obtidos utilizando 50% e 60% dos dados na fase de treinamento não foi estatisticamente relevante na maioria dos casos (segundo o teste *t de Student* bicaudal com 95% de confiança). Todos os resultados são indicados graficamente nas Figuras 84 a 86.

Pode-se notar nas Figuras 84 a 86 que os melhores resultados, para todos os casos testados, foram obtidos utilizando uma rede neural com 8 neurônios na camada intermediária. Também é possível observar que os melhores resultados, independente do tamanho dos conjuntos de treinamento, validação e teste, foram obtidos utilizando os conjuntos de dados Indiv_20, Indiv_40 e Indiv_50. Além disso, o desempenho das redes treinadas a partir desses conjuntos de dados foi bem semelhante nos casos onde foram utilizados 50% e 60% dos dados na fase de treinamento.

Tabela 27. Resultados obtidos utilizando a rede com duas saídas (*TO*).

Caso	Erro de Treinamento		Erro de Validação		Erro de Teste		Épocas	Tempo (s)
	μ	σ	μ	σ	μ	σ		
1	1,870e-03	2,533e-03	8,687e-03	1,049e-02	5,559e-03	2,956e-03	12	1,186
2	1,060e-03	1,754e-03	1,152e-02	1,450e-02	6,936e-03	2,220e-03	7	0,934
3	1,352e-03	1,663e-03	5,199e-03	8,654e-03	**2,981e-03**	2,575e-03	15	1,615
4	8,348e-04	1,426e-03	5,690e-03	9,110e-03	3,721e-03	2,494e-03	12	1,555
5	1,597e-03	2,211e-03	7,810e-03	1,178e-02	**2,652e-03**	1,798e-03	18	1,805
6	1,994e-03	2,147e-03	2,518e-03	2,046e-03	3,221e-03	1,563e-03	15	1,856
7	9,165e-04	1,046e-03	4,009e-03	6,112e-03	1,275e-03	8,615e-04	23	1,257
8	1,091e-03	1,771e-03	3,526e-03	4,980e-03	2,077e-03	1,594e-03	21	1,317
9	3,153e-04	2,849e-04	2,362e-03	3,916e-03	**3,903e-04**	1,526e-04	23	1,280
10	4,246e-04	4,236e-04	1,255e-03	1,528e-03	4,528e-04	1,412e-04	18	1,207
11	4,596e-04	3,607e-04	1,411e-03	2,893e-03	**2,687e-04**	1,533e-04	17	1,134
12	4,503e-04	5,471e-04	2,059e-03	2,449e-03	3,079e-04	1,335e-04	23	1,574
13	1,139e-03	1,742e-03	2,968e-03	3,408e-03	2,209e-03	1,615e-03	21	1,209
14	1,140e-03	1,685e-03	2,032e-03	2,510e-03	2,505e-03	1,432e-03	23	1,430
15	3,090e-04	2,331e-04	1,301e-03	1,586e-03	**6,996e-04**	1,199e-04	24	2,298
16	1,420e-04	6,282e-05	1,942e-03	3,093e-03	7,324e-04	1,295e-04	27	1,732
17	1,834e-04	8,087e-05	9,383e-04	1,106e-03	**5,401e-04**	1,075e-04	30	1,702
18	2,705e-04	4,379e-04	8,653e-04	1,113e-03	5,913e-04	8,295e-05	25	1,672
19	1,312e-03	1,678e-03	1,528e-03	1,859e-03	1,021e-03	8,697e-04	19	1,260
20	1,240e-03	1,023e-03	2,635e-03	3,053e-03	1,152e-03	8,441e-04	17	1,319
21	4,647e-04	3,126e-04	9,351e-04	1,330e-03	**3,765e-04**	1,198e-04	21	1,386
22	5,182e-04	4,661e-04	1,212e-03	1,321e-03	4,076e-04	1,155e-04	17	1,402
23	6,127e-04	4,414e-04	1,152e-03	1,202e-03	**3,393e-04**	8,926e-05	18	1,277
24	2,234e-04	1,777e-04	1,432e-03	1,624e-03	3,462e-04	8,047e-05	27	1,973
25	6,114e-04	6,984e-04	1,566e-03	1,726e-03	3,874e-04	1,940e-04	22	1,664
26	4,976e-04	5,286e-04	1,381e-03	1,749e-03	5,134e-04	2,960e-04	21	3,161
27	5,665e-04	6,305e-04	1,071e-03	1,485e-03	**2,681e-04**	1,178e-04	28	2,166
28	3,170e-04	2,669e-04	7,839e-04	1,044e-03	2,735e-04	9,378e-05	25	4,022
29	3,790e-04	3,737e-04	9,850e-04	1,213e-03	**2,378e-04**	5,798e-05	24	2,023
30	4,083e-04	5,293e-04	1,220e-03	1,209e-03	2,511e-04	7,709e-05	22	3,549

Uma vez que os indivíduos na população do PSO são classificados de acordo com o valores de *fitness*, para se utilizar uma rede baseada nesse modelo durante o processo de otimização, é necessário utilizar as equações (70) a (73) a fim de se obter esses valores. Dessa forma, utiliza-se os valores de *offset* máximo e tração no topo máxima aproximados pela rede neural para calcular os valores de *fitness*.

A partir dos valores calculados de *fitness*, verifica-se o erro ocorrido na aproximação desses valores com relação aos obtidos na simulação com análise estática. Esse erro foi denominado *Erro de Fitness*.

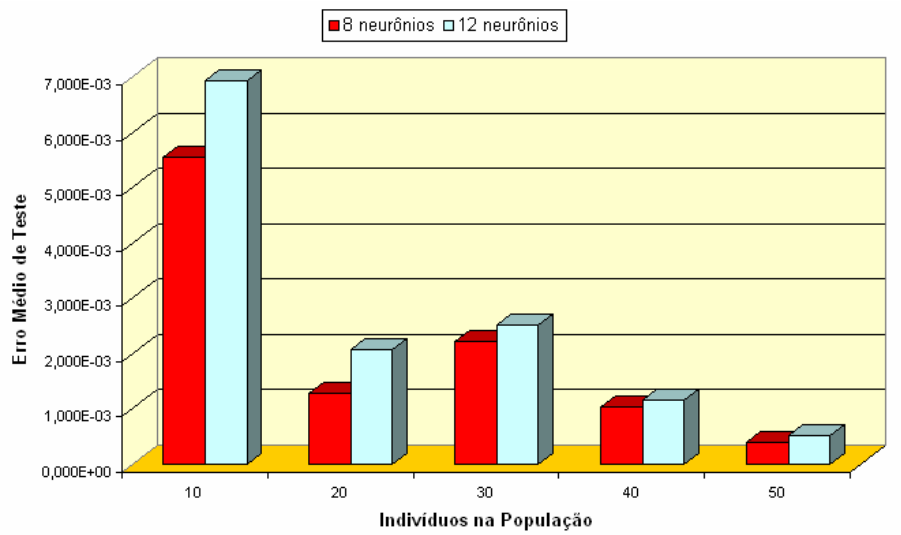

Figura 84. Resultados obtidos com 40% dos dados na fase de treinamento (*TO*).

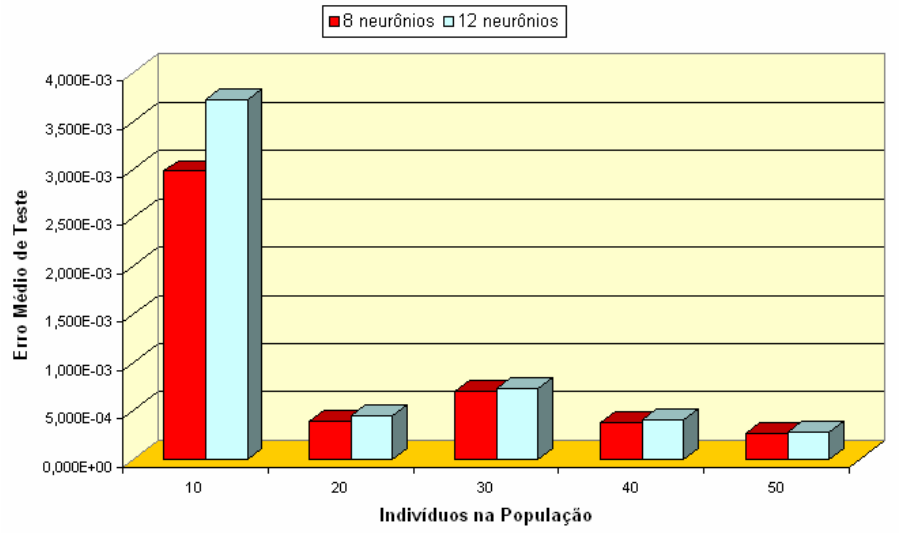

Figura 85. Resultados obtidos com 50% dos dados na fase de treinamento (*TO*).

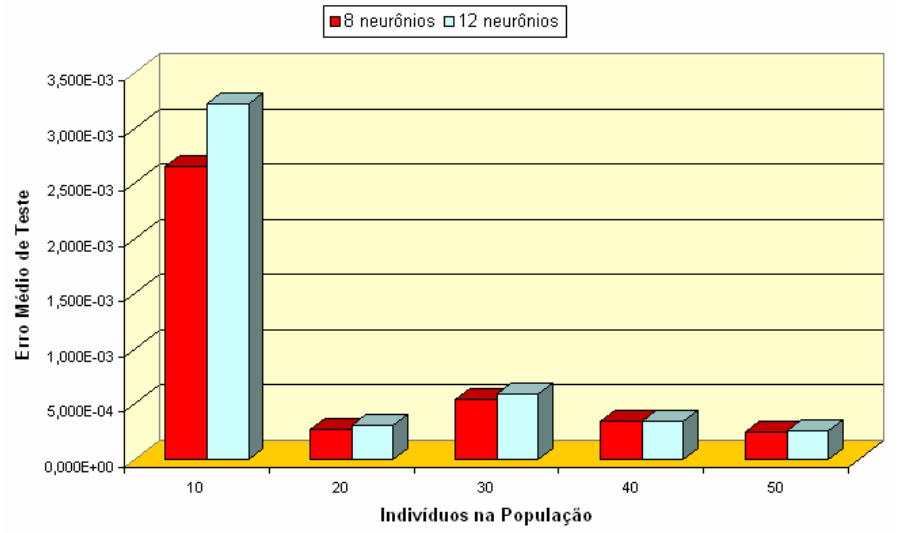

Figura 86. Resultados obtidos com 60% dos dados na fase de treinamento (*TO*).

A fim de avaliar o desempenho da rede na aproximação de cada valor de saída, verificou-se o erro ocorrido em cada uma das saídas da rede separadamente. Tais erros foram denominados *Erro de Offset* e *Erro de Tração*, relativos aos erros ocorridos na aproximação dos valores de *offset* máximo e de tração no topo máxima, respectivamente. Os resultados estatísticos dos experimentos em termos de média do *Erro de Offset*, do *Erro de Tração* e do *Erro de Fitness* (calculado) podem ser vistos na Tabela 28.

Tabela 28. Erros médios separados por parâmetro de saída do método *TO*.

Caso	***Erro de Offset***	***Erro de Tração***	***Erro de Fitness*** **(Calculado)**
1	4,507e-03	6,611e-03	2,507e-02
2	7,269e-03	6,604e-03	2,274e-02
3	3,264e-03	2,698e-03	1,113e-02
4	4,364e-03	3,079e-03	1,366e-02
5	1,750e-03	3,554e-03	**6,070e-03**
6	3,562e-03	2,880e-03	1,253e-02
7	9,286e-04	1,622e-03	2,988e-03
8	1,860e-03	2,294e-03	5,994e-03
9	8,272e-05	6,978e-04	**4,611e-04**
10	1,370e-04	7,686e-04	6,143e-04
11	1,406e-04	3,968e-04	5,261e-04
12	1,307e-04	4,852e-04	**4,783e-04**
13	1,628e-03	2,791e-03	4,314e-03
14	1,977e-03	3,033e-03	5,195e-03
15	7,332e-05	1,326e-03	**5,915e-04**
16	7,229e-05	1,392e-03	7,168e-04
17	**3,808e-05**	1,042e-03	**1,144e-04**
18	5,070e-05	1,132e-03	1,901e-04
19	8,791e-04	1,164e-03	3,811e-03
20	1,182e-03	1,122e-03	3,453e-03
21	1,378e-04	6,151e-04	**3,990e-04**
22	2,248e-04	5,905e-04	6,417e-04
23	1,510e-04	5,276e-04	4,532e-04
24	1,254e-04	5,671e-04	**3,843e-04**
25	1,989e-04	5,760e-04	8,442e-04
26	4,432e-04	5,836e-04	1,541e-03
27	1,408e-04	3,954e-04	4,344e-04
28	1,125e-04	4,345e-04	**3,574e-04**
29	6,212e-05	4,136e-04	**2,079e-04**
30	8,487e-05	4,174e-04	3,303e-04

Na Tabela 28 pode-se ver que o menor erro médio no valor de *fitness* calculado ocorreu no caso 17, provavelmente devido ao fato de ter obtido o menor erro médio no valor de *offset* máximo (que influencia diretamente no valor calculado da *fitness*). Embora a diferença entre os resultados obtidos utilizando 50% e 60% dos dados na fase de treinamento tenha sido relevante entre os casos 15 e 17, entre os casos 9 e 12, 21 e 24, e 28 e 29 isso não ocorreu. Os testes utilizando os dados obtidos a partir da otimização com 10 indivíduos na população (Indiv_10) obtiveram os 6 piores resultados entre todos os casos analisados., podendo-se concluir que o espaço de busca de soluções não foi suficientemente (ou apropriadamente) explorado nesse caso.

A Figura 87 mostra graficamente os erros médios de *offset*, tração e *fitness* (calculado) para cada um dos casos estudados. Pode-se observar que os casos utilizando o conjunto de dados Indiv_30 e 60% dos dados na fase de treinamento (casos 17 e 18), embora tenham obtido os valores mais baixos de erro no valor da *fitness* (1,144e-04 e 1,901e-04, respectivamente), apresentam um erro no valor de tração máxima maior do que os obtidos nos casos utilizando os conjuntos de dados Indiv_20, Indiv_40 e Indiv_50 com 50% ou 60% dos dados na fase de treinamento. Isso ocorre porque o valor de tração só influencia o valor de *fitness* se for inferior a 30% da MBL (Equação (71)) ou ultrapassar 50% da MBL (Equação (72)), incluindo, dessa maneira, valores de penalidade na função de *fitness* (Equação (73)). Uma vez que os valores de tração estejam na faixa entre 30% e 50% da MBL e o valor aproximado pela rede permaneça nessa faixa de valores, o erro encontrado na aproximação desse valor de tração é irrelevante para o cálculo da *fitness*. No entanto, em alguns casos, o erro na aproximação dos valores de tração pode levar à penalização errônea de indivíduos ou à não penalização de indivíduos que seriam penalizados na análise rigorosa, acarretando um aumento do erro no valor de *fitness* calculado.

Considerando-se que seja desejado um ganho de 50% no tempo computacional do procedimento de otimização e um menor erro na aproximação dos valores de tração máxima, pode-se ver na Tabela 28 e na Figura 87 que, as redes treinadas a partir de 50% do conjunto de dados Indiv_20 obtiveram resultados melhores do que os obtidos a partir de 50% do conjunto de dados Indiv_30 e, como foi mostrado na Tabela 24, o valor de *fitness* da configuração "ótima" encontrada no processo de otimização que gerou o conjunto de dados Indiv_20 foi maior. Sendo assim, pode-se concluir que, utilizando o modelo de redes com duas saídas (*TO*), os dados gerados num processo de otimização com 20 indivíduos na população são suficientes para treinar as redes de forma que seja

alcançada uma boa aproximação dos valores de saída do sistema de ancoragem, mesmo quando são utilizados apenas 50% dos dados para treinamento e validação (representando 140 avaliações pela análise rigorosa). Entretanto, os valores ótimos de *fitness* encontrados nos processos de otimização com 40 e 50 indivíduos na população foram maiores do que o valor obtido na otimização com 20 indivíduos.

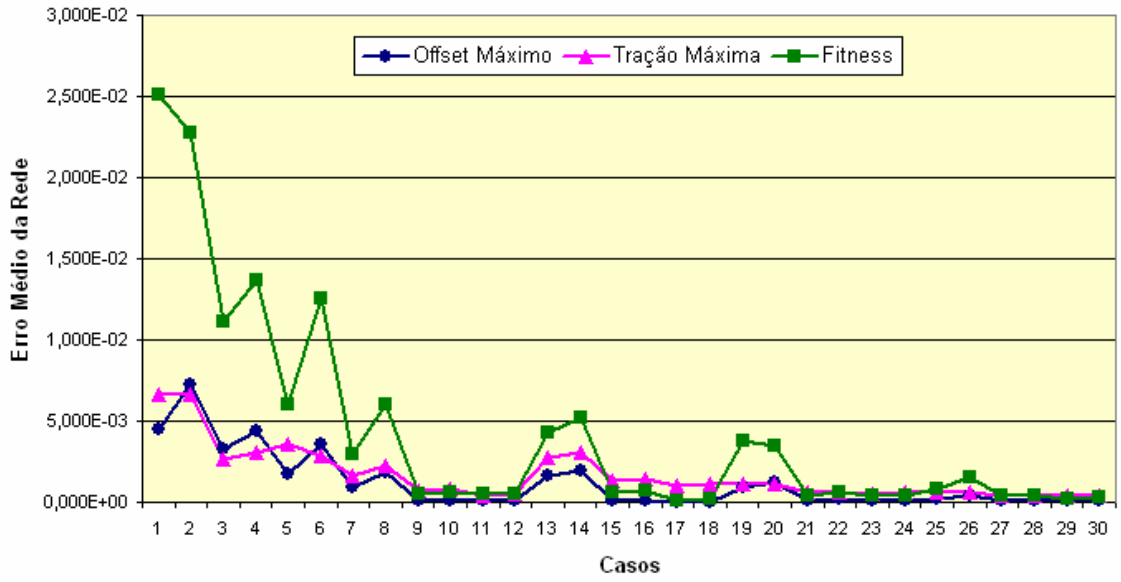

Figura 87. *Erro de Offset* x *Erro de Tração* x *Erro de Fitness* (calculado) (*TO*).

Caso seja necessário obter uma configuração melhor para o sistema de ancoragem (valor de *fitness* maior), embora sejam necessárias mais avaliações por meio da análise rigorosa, pode-se utilizar um processo de otimização com 40 indivíduos na população. Na Figura 87 é possível ver que o caso 21, que corresponde aos testes utilizando uma rede neural com 8 neurônios na camada intermediária e 50% dos dados do conjunto Indiv_40 na fase de treinamento, obteve bons resultados. Na verdade, devido ao modo como foram divididos inicialmente os conjuntos para treinamento, validação e teste, nesse caso, utilizou-se apenas 46% dos dados na fase de treinamento.

Uma vez que o treinamento completo da rede é muito rápido, correspondendo a um tempo menor do que o necessário para realizar apenas uma avaliação via análise estática, a aplicação desse método durante um processo de otimização pode representar uma economia de mais de 50% no tempo necessário para a otimização de projeto do sistema de ancoragem proposto.

Modelo 2 - Tração, Offset e Fitness como saídas para a Rede Neural (TOF)

Nesse segundo modelo de análise, foram utilizadas redes neurais com nove entradas e três saídas (*offset* máximo, tração no topo máxima e *fitness*). Os resultados estatísticos das realizações dos experimentos são apresentados na Tabela 29, em termos de média (μ) e desvio padrão (σ) dos erros obtidos nos conjuntos de treinamento, validação e teste, assim como o número de épocas e tempo médio necessários para o treinamento. Os valores de erro obtidos correspondem ao erro médio quadrático de todos os três valores de saída da rede. As Figuras 88 a 90 exibem graficamente os resultados obtidos.

Tabela 29. Resultados obtidos utilizando a rede com três saídas (*TOF*).

Caso	Erro de Treinamento		Erro de Validação		Erro de Teste		Épocas	Tempo (s)
	μ	σ	μ	σ	μ	σ		
1	2,221e-03	3,883e-03	1,054e-02	1,809e-02	6,205e-03	1,798e-03	13	0,857
2	1,848e-03	3,083e-03	5,910e-03	3,321e-03	7,475e-03	1,850e-03	13	0,908
3	3,569e-03	3,889e-03	5,871e-03	6,308e-03	5,030e-03	2,103e-03	17	1,045
4	3,300e-03	2,399e-03	8,038e-03	9,113e-03	6,277e-03	1,667e-03	14	0,892
5	1,727e-03	1,609e-03	3,238e-03	4,568e-03	**3,209e-03**	1,672e-03	23	1,213
6	1,695e-03	1,802e-03	6,862e-03	7,684e-03	4,402e-03	1,535e-03	16	1,059
7	7,160e-04	7,324e-04	3,393e-03	3,533e-03	2,564e-03	9,084e-04	24	1,324
8	8,024e-04	1,413e-03	4,757e-03	4,166e-03	3,595e-03	1,231e-03	25	1,632
9	3,210e-04	2,002e-04	1,458e-03	1,435e-03	4,093e-04	1,041e-04	27	1,633
10	3,548e-04	5,691e-04	2,121e-03	2,362e-03	6,494e-04	1,783e-04	29	1,847
11	3,494e-04	1,805e-04	2,040e-03	1,978e-03	**2,667e-04**	1,020e-04	25	1,570
12	2,781e-04	1,560e-04	1,357e-03	1,092e-03	3,910e-04	1,341e-04	24	1,788
13	4,678e-04	3,741e-04	1,495e-03	1,226e-03	1,699e-03	5,510e-04	27	1,460
14	1,267e-03	2,037e-03	3,321e-03	2,386e-03	2,531e-03	1,120e-03	21	1,416
15	1,661e-04	5,827e-05	6,621e-04	5,163e-04	5,722e-04	1,040e-04	38	2,982
16	1,186e-04	6,784e-05	9,102e-04	1,185e-03	6,566e-04	1,088e-04	36	2,578
17	1,472e-04	3,360e-05	1,004e-03	1,376e-03	**4,798e-04**	7,377e-05	41	3,414
18	1,235e-04	5,903e-05	7,318e-04	8,194e-04	5,184e-04	9,550e-05	39	2,863
19	3,254e-04	1,955e-04	1,204e-03	1,027e-03	6,525e-04	2,786e-04	28	1,719
20	3,515e-04	5,113e-04	1,177e-03	1,331e-03	9,103e-04	7,604e-04	30	2,282
21	3,966e-04	5,715e-04	1,224e-03	1,190e-03	3,074e-04	1,868e-04	27	1,833
22	1,980e-04	1,494e-04	1,035e-03	9,791e-04	3,559e-04	1,109e-04	33	2,586
23	3,472e-04	3,773e-04	1,376e-03	1,491e-03	**2,586e-04**	1,065e-04	32	2,234
24	2,007e-04	2,502e-04	1,003e-03	6,479e-04	3,083e-04	1,373e-04	36	2,991
25	1,080e-03	9,037e-04	1,624e-03	1,227e-03	7,415e-04	4,424e-04	26	1,944
26	5,479e-04	6,383e-04	1,752e-03	1,482e-03	9,618e-04	1,182e-03	29	2,595
27	3,747e-04	3,576e-04	1,050e-03	8,989e-04	2,396e-04	1,590e-04	35	2,920
28	3,259e-04	3,418e-04	8,322e-04	5,388e-04	2,658e-04	1,502e-04	28	2,956
29	3,813e-04	4,059e-04	7,633e-04	6,478e-04	**1,953e-04**	7,233e-05	32	2,872
30	2,140e-04	1,585e-04	5,696e-04	4,938e-04	1,985e-04	7,895e-05	35	3,873

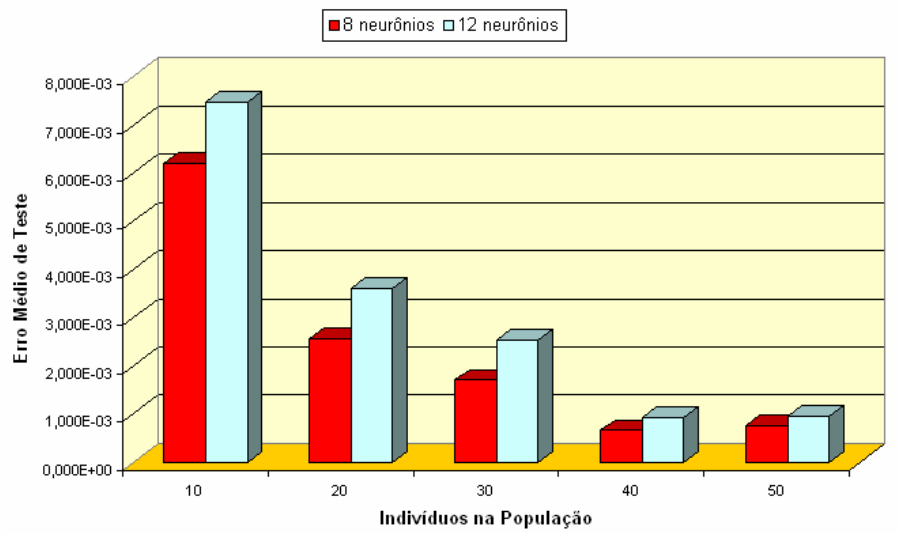

Figura 88. Resultados obtidos com 40% dos dados na fase de treinamento (*TOF*).

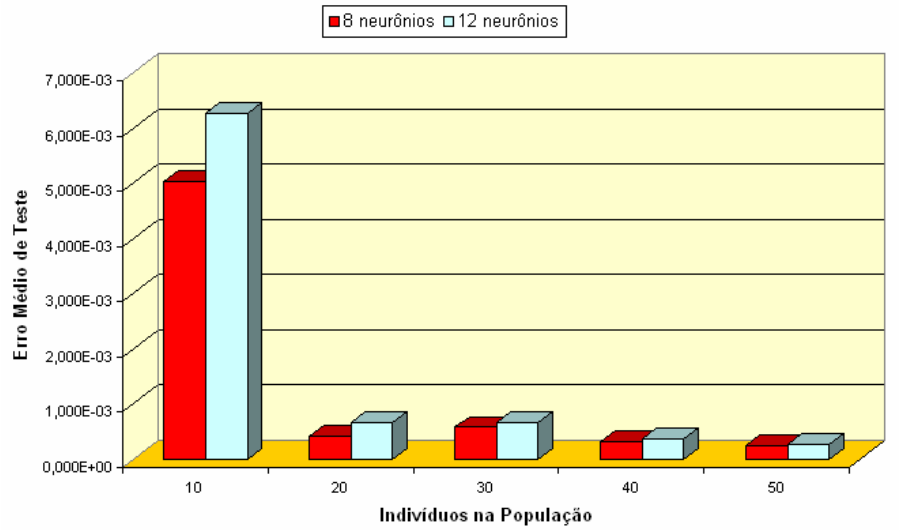

Figura 89. Resultados obtidos com 50% dos dados na fase de treinamento (*TOF*).

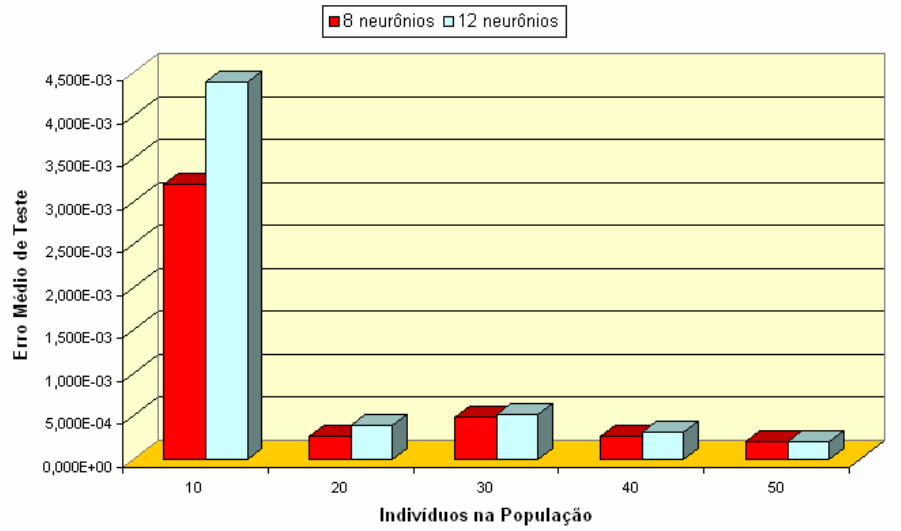

Figura 90. Resultados obtidos com 60% dos dados na fase de treinamento (*TOF*).

Pode-se observar na Tabela 29 que, assim como no primeiro modelo, a diferença entre os resultados obtidos utilizando 50% e 60% dos dados na fase de treinamento não foi estatisticamente relevante.

A partir das Figuras 88 a 90 pode-se notar que as redes neurais com 8 neurônios na camada intermediária novamente obtiveram os melhores resultados em todos os casos testados. É possível observar que as redes treinadas utilizando os conjuntos de dados Indiv_40 e Indiv_50 alcançaram resultados semelhantes em cada um dos casos considerados de porcentagem do conjunto para a fase de treinamento. No entanto, como se trata de porcentagem do tamanho total do conjunto de dados obtido na otimização, foram necessários menos exemplos para o treinamento das redes que utilizaram o conjunto de dados Indiv_40. Nos casos onde utilizou-se 40% dos dados na fase de treinamento, as redes treinadas utilizando os conjuntos de dados Indiv_20 e Indiv_30 obtiveram resultados significativamente piores do que os obtidos utilizando os conjuntos de dados Indiv_40 e Indiv_50. Novamente, os testes utilizando o conjunto de dados Indiv_10 obtiveram os piores resultados.

Diferentemente do primeiro modelo proposto, que utiliza apenas *offset* e tração máximos como saída para a rede, nesse segundo modelo não é necessário nenhum cálculo posterior para encontrar os valores de *fitness* que serão utilizados para classificar os indivíduos na população do PSO. Como o valor de *fitness* é uma das variáveis de saída da rede neural, a fim de verificar qual caso estudado obteve o melhor desempenho na aproximação desses valores, verificou-se o erro ocorrido em cada uma das saídas da rede separadamente. Analogamente à análise do modelo anterior, tais erros foram denominados *Erro de Offset*, *Erro de Tração* e *Erro de Fitness*, relativos aos erros ocorridos na aproximação dos valores de *offset* máximo, de tração no topo máxima e de *fitness*, respectivamente. Os resultados estatísticos dos experimentos em termos de média do *Erro de Offset*, do *Erro de Tração* e do *Erro de Fitness* podem ser vistos na Tabela 30.

Pode-se ver na Tabela 30 que, assim como ocorreu no primeiro modelo, o menor erro médio no valor de *fitness* ocorreu no caso 17, que utiliza uma rede neural com 8 neurônios na camada intermediária treinada com 60% dos dados do conjunto Indiv_30. Embora a diferença entre os resultados obtidos utilizando 50% e 60% dos dados na fase de treinamento tenha sido relevante entre os casos 15 e 17, nos demais casos, isso não ocorreu.

Tabela 30. Erros médios separados por parâmetro de saída do método *TOF*.

Caso	Erro de Offset	Erro de Tração	Erro de Fitness
1	4,328e-03	5,957e-03	8,331e-03
2	3,323e-03	7,866e-03	1,124e-02
3	6,201e-03	3,755e-03	5,134e-03
4	7,233e-03	4,370e-03	7,230e-03
5	3,399e-03	2,761e-03	**3,468e-03**
6	5,057e-03	3,131e-03	5,017e-03
7	1,025e-03	1,842e-03	4,824e-03
8	1,196e-03	3,132e-03	6,458e-03
9	4,274e-05	6,354e-04	5,498e-04
10	1,968e-04	1,022e-03	7,297e-04
11	3,387e-05	4,150e-04	**3,513e-04**
12	8,695e-05	4,841e-04	6,019e-04
13	3,547e-04	3,149e-03	1,594e-03
14	1,434e-03	3,275e-03	2,884e-03
15	3,074e-05	1,297e-03	3,884e-04
16	5,128e-05	1,537e-03	3,816e-04
17	5,513e-05	1,320e-03	**6,454e-05**
18	3,200e-05	1,437e-03	8,660e-05
19	9,530e-05	8,689e-04	9,933e-04
20	5,483e-04	1,110e-03	1,072e-03
21	8,500e-05	6,542e-04	1,830e-04
22	8,244e-05	8,462e-04	1,390e-04
23	8,274e-05	5,712e-04	**1,217e-04**
24	9,000e-05	7,561e-04	7,870e-05
25	3,094e-04	1,144e-03	7,716e-04
26	4,299e-04	1,785e-03	6,709e-04
27	7,305e-05	4,629e-04	1,828e-04
28	8,169e-05	5,364e-04	1,793e-04
29	3,637e-05	4,059e-04	**1,437e-04**
30	3,836e-05	4,444e-04	1,126e-04

A Figura 91 mostra graficamente os erros médios de *offset*, tração e *fitness* para cada um dos casos estudados. Pode-se ver que, na maioria dos casos, o erro no valor de *offset* foi o menor dentre os erros dos três valores de saída da rede.

A partir da Tabela 30 e da Figura 91 pode-se observar que os melhores resultados foram obtidos nos casos que utilizam redes neurais treinadas com 60% dos dados dos conjuntos Indiv_30, Indiv_40 e Indiv_50. No entanto, nos casos que utilizam os conjuntos de dados Indiv_40 e Indiv_50, a diferença entre os resultados obtidos utilizando 50% e 60% dos dados na fase de treinamento não foi relevante.

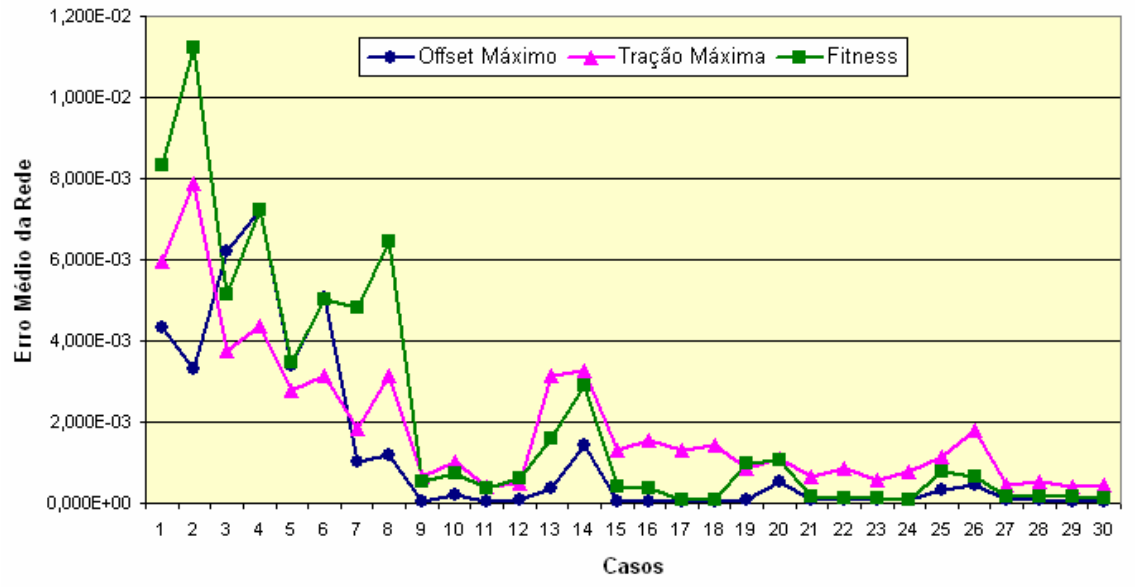

Figura 91. *Erro de Offset* x *Erro de Tração* x *Erro de Fitness* (*TOF*).

Pode-se concluir que, utilizando o modelo de redes com três saídas (*TOF*), 60% dos dados gerados num processo de otimização com 30 indivíduos na população são capazes de fornecer informação necessária para treinar uma rede neural alcançando um baixo erro médio na aproximação dos valores de *fitness* (6,454e-05).

Contudo, considerando-se que seja desejada a obtenção de uma melhor configuração para o sistema de ancoragem proposto (ou seja, com valor de *fitness* mais elevado), embora sejam necessárias mais avaliações por meio da análise rigorosa, pode-se utilizar um processo de otimização com 40 indivíduos na população. Na Tabela 30 pode-se ver que o caso 22, que corresponde aos testes utilizando uma rede neural com 12 neurônios na camada intermediária e 50% dos dados do conjunto Indiv_40 na fase de treinamento, obteve bons resultados.

Comparando-se as Tabelas 28 e 30 pode-se observar que, o treinamento das redes neurais baseadas nesse segundo modelo (*TOF*) obteve erros médios no valor de *offset* menores do que os obtidos pelo modelo *TO* em 73% dos casos testados. Além disso, não sendo necessário o cálculo posterior dos valores de *fitness* e, uma vez que o tempo adicional necessário para o treinamento foi de, no máximo, 1,712s (no caso 17), a aplicação desse método durante um processo de otimização pode representar uma economia ainda maior no tempo necessário para a otimização de projeto do sistema de ancoragem proposto. Uma comparação mais detalhada dos métodos será apresentada mais adiante.

Modelo 3 - Fitness como saída para a Rede Neural (F)

Baseado no fato de que, durante o processo de otimização do PSO, o único parâmetro de resposta do sistema que interfere no processo de evolução dos indivíduos é o valor de *fitness*, nesse terceiro modelo de análise, foram utilizadas redes neurais com nove entradas e apenas uma saída (*fitness*). Os resultados estatísticos das realizações dos experimentos são apresentados na Tabela 31, em termos de média (μ) e desvio padrão (σ) dos erros obtidos nos conjuntos de treinamento, validação e teste, assim como o número de épocas e tempo médio necessários para o treinamento. As Figuras 92 a 94 exibem graficamente os resultados obtidos.

Tabela 31. Resultados obtidos utilizando a rede com apenas uma saída (*F*).

Caso	Erro de Treinamento		Erro de Validação		Erro de Teste		Épocas	Tempo (s)
	μ	σ	μ	σ	μ	σ		
1	4,352e-03	3,921e-03	7,514e-03	8,209e-03	1,201e-02	4,765e-03	10	0,633
2	5,222e-03	5,054e-03	1,873e-02	1,395e-02	1,505e-02	3,479e-03	6	0,548
3	2,992e-03	2,706e-03	4,860e-03	4,381e-03	4,749e-03	1,580e-03	12	0,717
4	3,948e-03	2,984e-03	5,967e-03	4,217e-03	5,403e-03	1,096e-03	10	0,684
5	3,485e-03	5,154e-03	4,583e-03	2,846e-03	**4,005e-03**	1,763e-03	13	0,791
6	3,639e-03	4,617e-03	6,289e-03	4,321e-03	4,884e-03	1,428e-03	12	0,796
7	2,299e-03	2,234e-03	4,137e-03	2,407e-03	7,145e-03	3,612e-03	13	0,938
8	2,910e-03	2,483e-03	5,572e-03	2,678e-03	9,405e-03	3,641e-03	13	0,829
9	1,360e-03	1,126e-03	3,192e-03	2,529e-03	1,437e-03	9,849e-04	17	0,928
10	1,730e-03	8,815e-04	4,286e-03	2,247e-03	2,892e-03	1,989e-03	14	0,880
11	1,398e-03	9,198e-04	2,045e-03	1,209e-03	**8,371e-04**	2,997e-04	24	1,176
12	2,432e-03	1,482e-03	3,377e-03	1,525e-03	1,463e-03	9,472e-04	14	0,931
13	2,470e-03	1,668e-03	3,822e-03	2,502e-03	3,335e-03	1,619e-03	17	0,886
14	2,806e-03	1,468e-03	5,946e-03	2,905e-03	5,644e-03	2,078e-03	13	0,838
15	1,691e-05	7,206e-06	2,327e-04	3,255e-04	**2,502e-04**	1,473e-04	38	1,734
16	2,028e-05	1,992e-05	3,546e-04	3,675e-04	3,673e-04	2,085e-04	37	1,919
17	2,491e-04	7,041e-04	5,815e-04	1,558e-03	**1,267e-04**	1,989e-04	44	1,890
18	2,853e-04	8,127e-04	4,091e-04	6,473e-04	1,620e-04	1,788e-04	37	1,902
19	1,480e-04	3,687e-04	7,898e-04	9,498e-04	1,200e-03	1,089e-03	32	1,482
20	6,338e-04	1,131e-03	1,165e-03	1,838e-03	1,303e-03	7,554e-04	32	1,717
21	1,816e-04	4,945e-04	7,496e-04	6,692e-04	**1,392e-04**	2,062e-04	40	1,843
22	2,127e-04	4,990e-04	1,106e-03	1,449e-03	1,652e-04	1,604e-04	32	1,813
23	8,492e-05	1,227e-04	3,196e-04	3,621e-04	**3,452e-05**	3,180e-05	35	1,805
24	4,127e-05	5,332e-05	6,229e-04	7,603e-04	5,585e-05	5,092e-05	40	2,252
25	3,108e-04	5,998e-04	7,673e-04	9,417e-04	4,252e-04	5,672e-04	34	1,803
26	2,180e-04	3,313e-04	1,061e-03	1,444e-03	4,377e-04	3,909e-04	34	2,010
27	2,155e-04	2,954e-04	5,576e-04	4,686e-04	**7,069e-05**	7,005e-05	32	1,822
28	2,380e-04	4,409e-04	6,292e-04	6,328e-04	1,338e-04	1,486e-04	31	2,035
29	1,084e-04	1,259e-04	4,970e-04	4,878e-04	**4,983e-05**	5,643e-05	34	2,050
30	1,353e-04	1,269e-04	5,777e-04	4,558e-04	6,348e-05	5,129e-05	41	2,680

Nas Figuras 92 a 94 pode-se notar que, novamente, as redes com 8 neurônios na camada intermediária obtiveram os melhores resultados em todos os casos testados.

Na Tabela 31 é possível observar que o caso 23, que utiliza uma rede neural com 8 neurônios na camada intermediária treinada com 60% dos dados do conjunto Indiv_40, obteve os melhores resultados de média e desvio padrão do erro no conjunto de teste. Como a rede neural utilizada possui apenas uma saída (*fitness*), a média do erro no conjunto de teste equivale ao *Erro de Fitness* dos modelos anteriores.

É possível observar nas Figuras 92 a 94 que esse modelo obteve resultados significativamente piores quando foram utilizados os conjuntos de dados Indiv_10 e Indiv_20 no treinamento das redes. Isso sugere que, a partir das primeiras gerações do processo de otimização com 10 ou 20 indivíduos na população, não foi possível conseguir informações suficientes acerca da influência dos valores de entrada sobre o valor de *fitness*. Novamente, as redes neurais treinadas utilizando o conjunto de dados Indiv_10 obtiveram os 6 piores resultados entre todos os casos analisados.

Nos casos onde foi utilizado 50% dos dados na fase de treinamento, a rede com 8 neurônios na camada intermediária, treinada utilizando o conjunto de dados Indiv_50, obteve o melhor resultado na aproximação do valor de *fitness* (caso 27). No entanto, nesse caso, foi necessário avaliar 400 configurações candidatas a partir da análise estática a fim de construir o conjunto de dados utilizado na fase de treinamento. Sendo assim, o caso 21, que utiliza uma rede neural com 8 neurônios na camada intermediária treinada com 50% dos dados do conjunto Indiv_40, pode constituir uma alternativa mais eficiente, uma vez que são necessárias 240 avaliações de configurações candidatas via análise estática.

Embora o processo de otimização utilizando 30 indivíduos na população do PSO tenha convergido para uma solução cujo valor de *fitness* foi o segundo pior dentre os experimentos realizados, levando-se em consideração o desempenho computacional do procedimento, pode-se sugerir a utilização de uma rede neural com 8 neurônios na camada intermediária treinada com 50% dos dados do conjunto Indiv_30 (caso 15), onde foram necessárias 180 avaliações por análise estática obtendo um erro médio de 2,502e-04 no conjunto de teste.

É importante ressaltar que, ao final de um processo de otimização utilizando uma rede neural construída baseada nesse modelo, será necessário realizar uma análise estática da configuração considerada "ótima" a fim de obter os valores de *offset* máximo e tração no topo máxima do sistema de ancoragem otimizado.

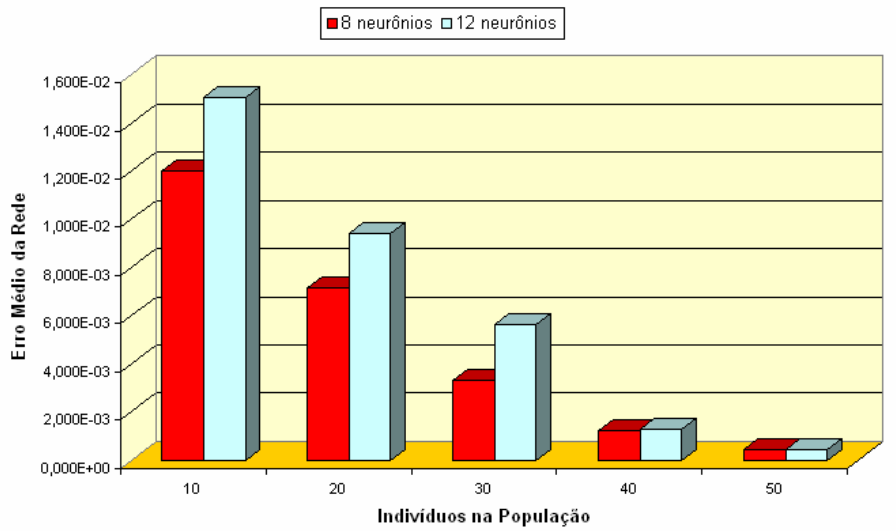

Figura 92. Resultados obtidos com 40% dos dados na fase de treinamento (*F*).

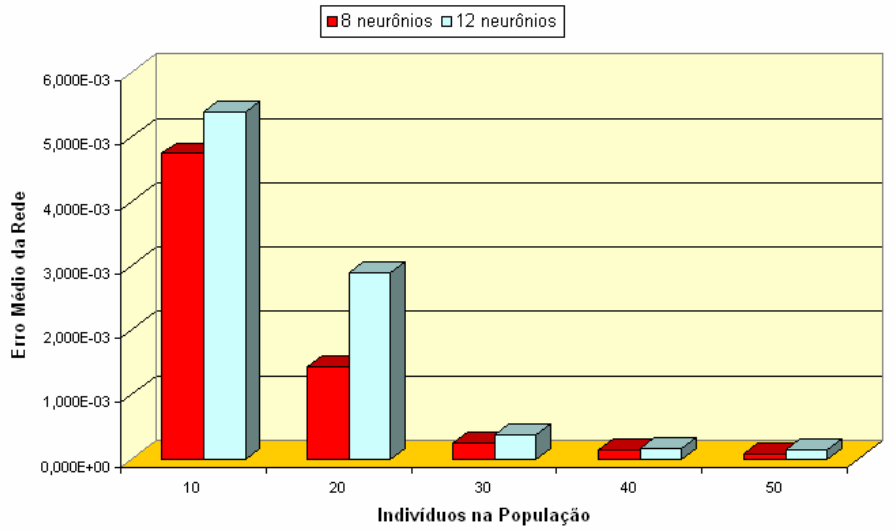

Figura 93. Resultados obtidos com 50% dos dados na fase de treinamento (*F*).

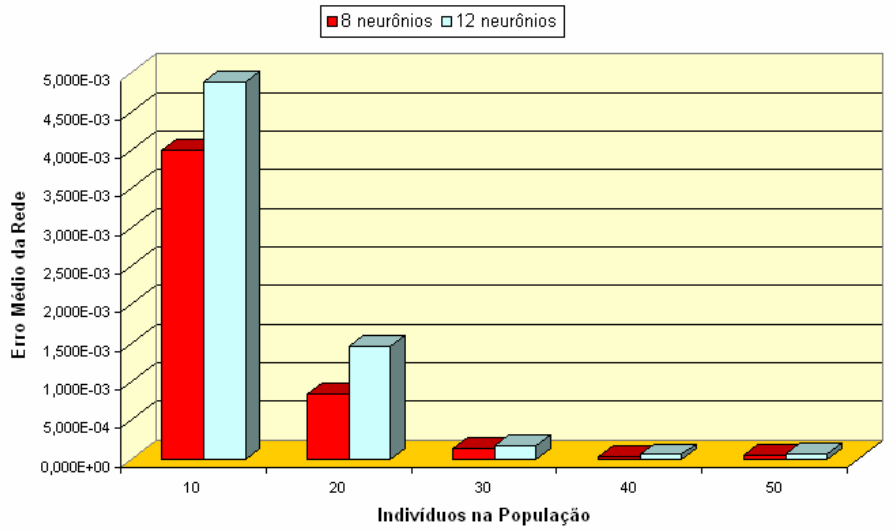

Figura 94. Resultados obtidos com 60% dos dados na fase de treinamento (*F*).

Comparação dos Modelos

Uma vez que, durante a otimização, o único parâmetro de resposta do sistema que interfere no processo de evolução dos indivíduos na população do PSO é o valor de *fitness*, utilizou-se os resultados obtidos na aproximação desse valor como parâmetro de comparação entre os modelos propostos.

As Figuras 95 a 99 apresentam graficamente os resultados de *Erro de Fitness* obtidos nos experimentos aplicando-se os três modelos analisados em cada caso estudado.

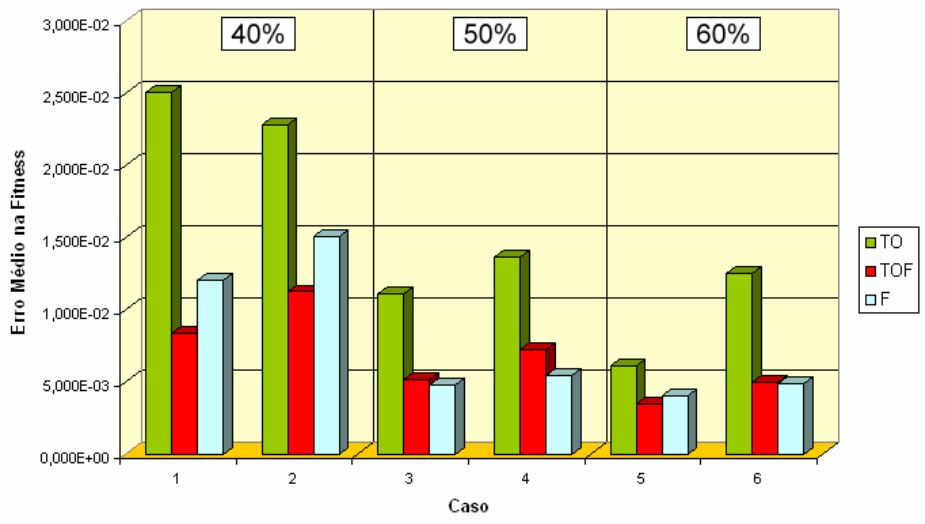

Figura 95. (*TO* x *TOF* x *F*): *Erros de Fitness* obtidos com Indiv_10.

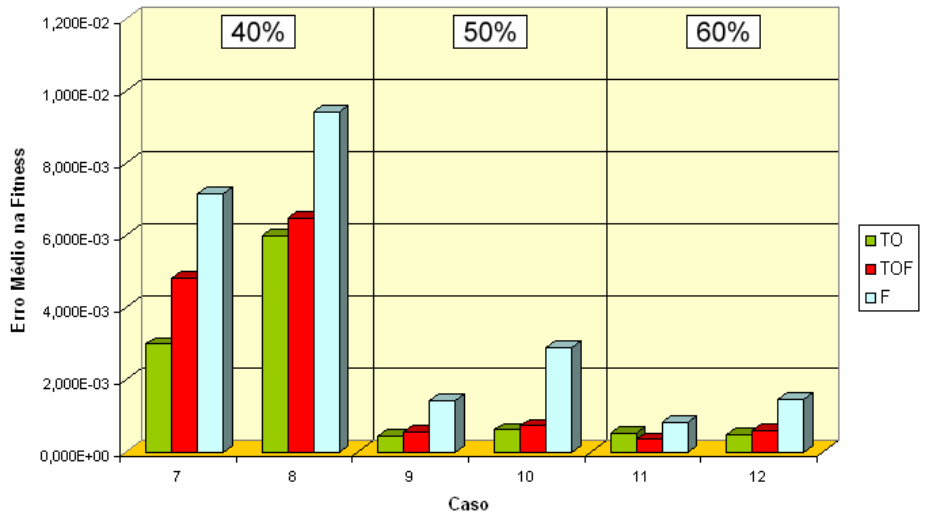

Figura 96. (*TO* x *TOF* x *F*): *Erros de Fitness* obtidos com Indiv_20.

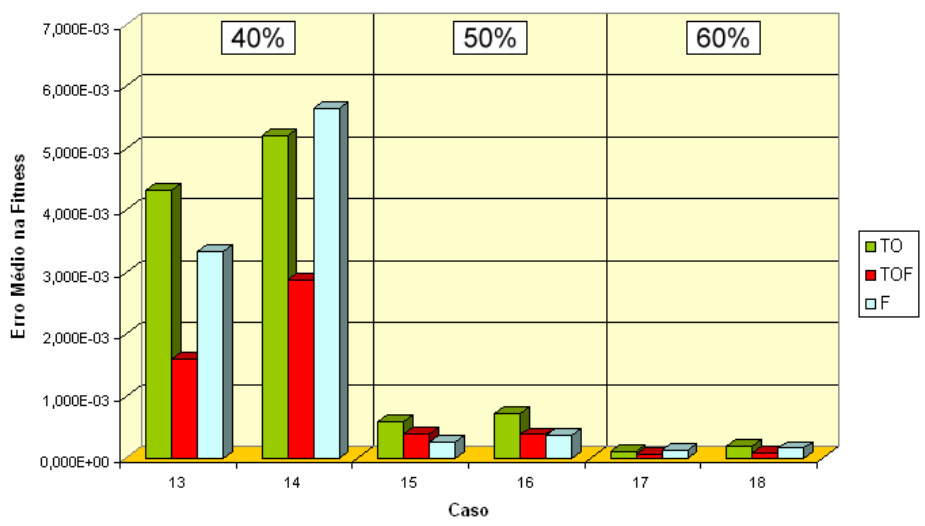

Figura 97. (*TO* x *TOF* x *F*): *Erros de Fitness* obtidos com Indiv_30.

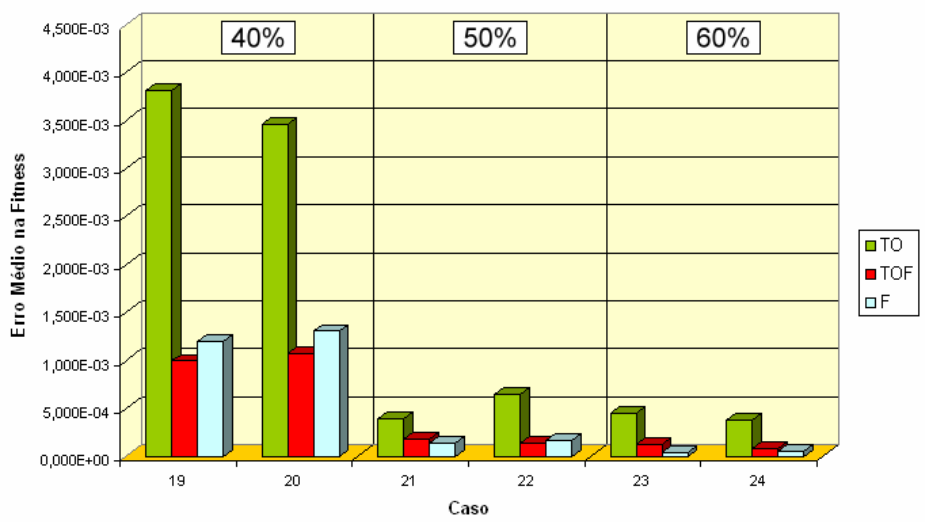

Figura 98. (*TO* x *TOF* x *F*): *Erros de Fitness* obtidos com Indiv_40.

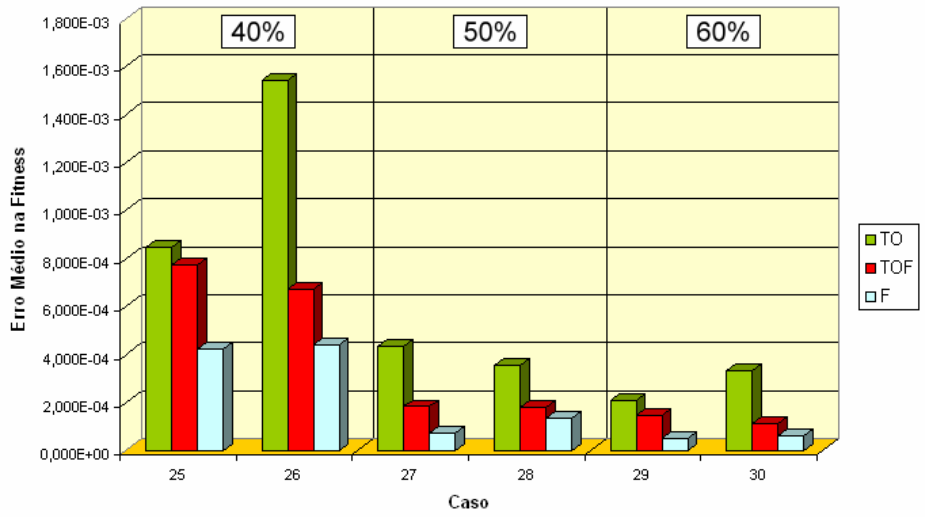

Figura 99. (*TO* x *TOF* x *F*): *Erros de Fitness* obtidos com Indiv_50.

Nas Figuras 95 a 99 é possível observar que os piores resultados obtidos ocorreram utilizando-se 40% dos dados para treinamento e validação da rede. No entanto, nos casos 25 e 26, que utilizaram o conjunto de dados Indiv_50 para o treinamento da rede, obteve-se erros médios no valor de *fitness* iguais a 4,252e-4 e 4,377e-4, respectivamente. Todavia, nesses casos, é necessário avaliar 300 configurações candidatas a partir da análise estática a fim de construir o conjunto de dados utilizado na fase de treinamento. Baseado no fato de que os erros obtidos nos casos 23 e 24, que utilizam redes neurais treinadas com 60% do conjunto de dados Indiv_40, foram de 3,452e-5 e 5,585e-5, respectivamente, sendo necessário avaliar 280 configurações via análise estática, conclui-se que não é recomendável a utilização de 50 indivíduos na população do processo de otimização uma vez que não representa ganho em eficiência e nem em desempenho computacional.

Considerando-se que um erro aceitável na aproximação do valor de *fitness* esteja, pelo menos, na ordem de 10^{-4}, conclui-se que utilizar apenas 10 indivíduos na população do PSO não é suficiente para treinar apropriadamente as redes, independente do modelo aplicado.

Na Figura 96 pode-se ver que o modelo utilizando apenas o valor de *fitness* como saída (*F*) obteve os piores resultados de aproximação nos casos utilizando os dados gerados na otimização com 20 indivíduos. No entanto, os resultados obtidos pelas redes baseadas no modelo *TOF* foram bem melhores, o que sugere que as informações adicionais de *offset* e tração nas saídas da rede, nesse caso, ajudaram no processo de treinamento, embora tenha sido necessário ajustar mais pesos na rede.

Com o objetivo de obter um ganho computacional maior no processo de otimização, pode-se considerar a utilização de 50% dos dados para treinamento e validação da rede. A partir das Figuras 95 a 99 é possível observar que, nesses casos, as redes neurais com 8 neurônios na camada intermediária obtiveram os melhores resultados.

Assim, a fim de verificar os resultados obtidos em cada um dos modelos nas 20 execuções de treinamento nos casos utilizando redes com 8 neurônios na camada intermediária, treinadas com 50% dos dados, foi utilizada a análise REC (*Regression Error Characteristic*) (BI & BENNETT, 2003).

A área sobre a curva REC (*Area Over Curve* – AOC) é uma estimativa do erro previsto para um modelo de regressão. Quanto menor o AOC, melhor a função de

regressão. Uma função de regressão domina outra se sua curva REC está sempre acima da curva REC que corresponde à outra função.

Utilizou-se no cálculo das curvas REC os valores médios aproximados, obtidos a partir das 20 execuções de treinamento realizadas. Sendo assim, o AOC será uma estimativa do erro médio.

Os gráficos REC gerados podem ser vistos na Figura 100. Os números entre parênteses nas figuras são a área sobre os valores da curva para cada curva REC. Analisando os valores AOC e a posição relativa das curvas REC, pode-se concluir que o modelo *F*, que utiliza apenas a *fitness* como saída, conseguiu o melhor desempenho na maioria dos casos, perdendo apenas no caso onde foi utilizado o conjunto de dados Indiv_20. Nas Figuras 100c e 100e pode-se ver que o modelo *F* dominou os outros modelos nos casos utilizando os conjuntos de dados Indiv_30 e Indiv_50 (note que a curva para *F* cobre as curva para *TO* e *TOF*). Sendo assim, pode-se concluir que o modelo *F* é preferível.

A fim de verificar a capacidade de aproximação global do metamodelo de redes neurais escolhido, utilizou-se as redes obtidas a partir das 20 execuções de treinamento do caso 21, que utiliza uma rede neural com 8 neurônios na camada intermediária treinada com 50% dos dados do conjunto Indiv_40, para aproximar os valores dos outros conjuntos de dados testados. Os resultados estatísticos das realizações dos experimentos são apresentados na Tabela 32, em termos de média (μ) e desvio padrão (σ) dos erros obtidos nos conjuntos.

Tabela 32. Resultados de erro de aproximação do metamodelo global.

Conjunto de Dados	**Erro de Aproximação**	
	μ	**σ**
Indiv_10	2,130E-04	3,812E-04
Indiv_20	1,400E-03	5,435E-04
Indiv_30	5,859E-04	3,053E-04
Indiv_50	1,504E-04	2,450E-04

Pode-se concluir pela Tabela 32 que as redes neurais obtidas foram capazes de aproximar satisfatoriamente os resultados da otimização (exceto no caso utilizando o conjunto de dados Indiv_20) e, portanto, a estratégia de metamodelagem com aproximação sequencial pode ser considerada uma boa alternativa na otimização de projeto do modelo de sistema de ancoragem proposto.

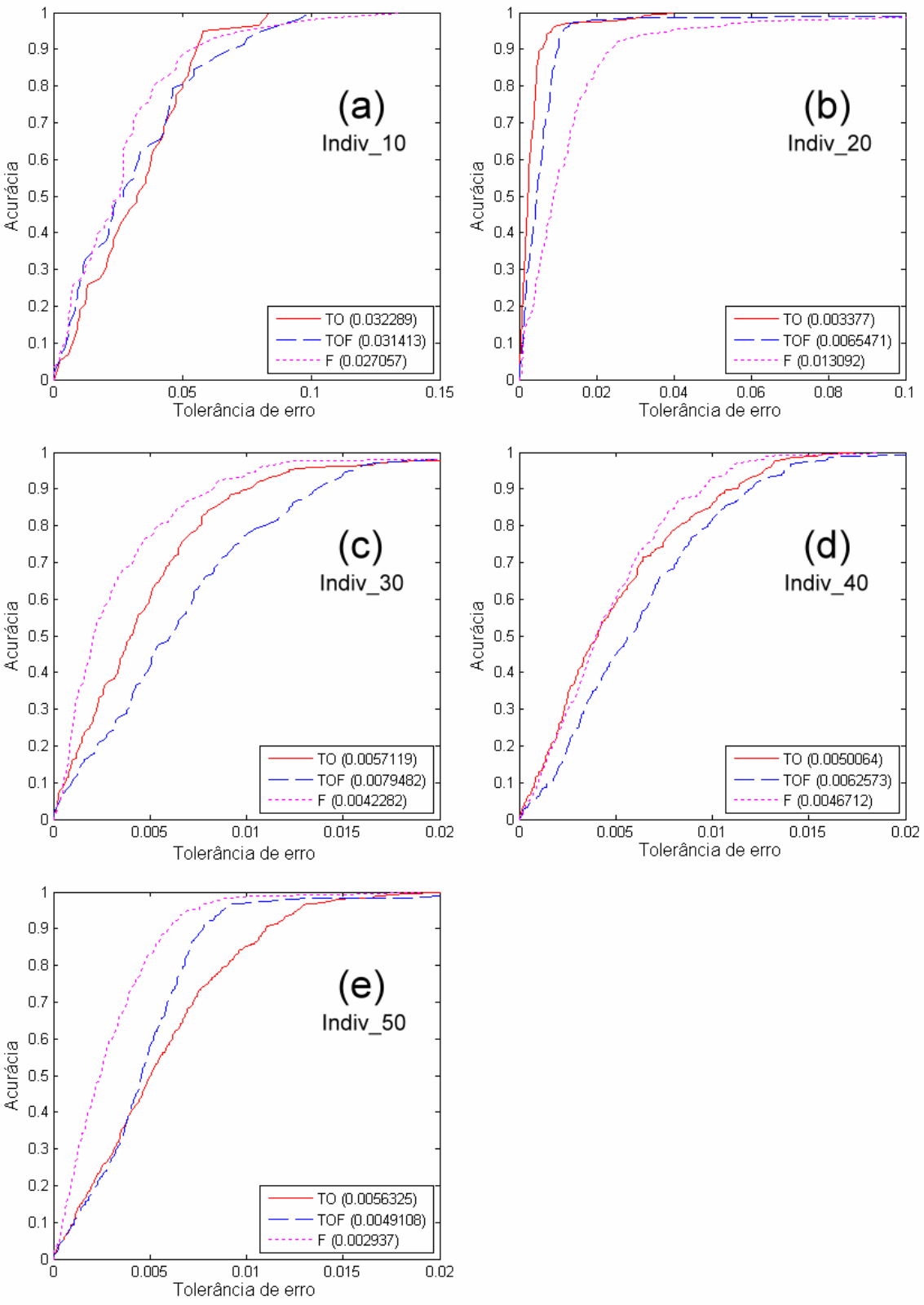

Figura 100. Gráficos REC de comparação dos modelos *TO*, *TOF* e *F*.

Como uma última observação, vale comentar que uma das redes obtidas nas 20 execuções de treinamento no caso 21, que utiliza uma rede neural com 8 neurônios na camada intermediária treinada com 50% dos dados do conjunto Indiv_40, conseguiu aproximar com exatidão o valor de *fitness* máximo (0,7645) obtido no processo de otimização por Enxame de Partículas. O valor médio de *fitness* obtido para a configuração "ótima" nas 20 execuções foi 0,7652, sendo que, independente do erro ocorrido, o valor encontrado continuou sendo o maior. Obviamente, não seria correto afirmar que, se essa rede neural fosse empregada durante o processo de otimização para substituir a análise rigorosa, a mesma configuração ótima seria encontrada. Uma vez que o valor de *fitness* é utilizado diretamente na atualização do melhor global e do melhor local de cada indivíduo, além de ser utilizado no cálculo do centro de massa da população (Equação (5)), qualquer erro ocorrido nesse valor se refletiria na atualização de velocidade (Equação (6)) e de posição (Equação (1)) das partículas no enxame, o que alteraria as configurações candidatas testadas no processo.

8.3 Conclusões

Neste capítulo foram apresentados dois exemplos de aplicação de redes neurais artificiais como metamodelos globais para processos de otimização.

Na aproximação da função DeJong F1, os resultados experimentais mostraram que uma rede neural treinada a partir de 120 exemplos obtidos durante um processo de otimização por Enxame de Partículas com 30 indivíduos na população é capaz de aproximar a função com um erro muito baixo (5,499e-06).

Na aplicação de redes neurais como metamodelos associados à otimização de projeto de sistemas de ancoragem, verificou-se que uma rede neural com apenas uma saída (*fitness*), treinada com 216 exemplos (validada com 24 exemplos) gerados durante um processo de otimização com 40 indivíduos na população foi capaz de aproximar diretamente a resposta da função de *fitness* do problema (função objetivo penalizada, com penalidades calculadas através de análise estática), com um erro de 1,392e-04. Como, no processo de otimização completo, foram executadas 520 avaliações da função, a substituição da análise rigorosa pela rede neural na otimização, depois do tempo de aquisição dos dados para treinamento, pode representar uma economia de 54% no tempo computacional necessário para otimizar um modelo de sistema de ancoragem.

9 COMENTÁRIOS FINAIS

9.1 Conclusões

Neste trabalho, foram apresentados estudos sobre três metodologias de otimização capazes de facilitar o desenvolvimento de projetos de estruturas *offshore* em diferentes etapas do processo. Para construir essas metodologias, foram empregadas duas ferramentas computacionais: o método de otimização por Enxame de Partículas (PSO), e Redes Neurais Artificiais (RNA) compondo metamodelos para substituir análises dinâmicas computacionalmente dispendiosas.

A primeira metodologia apresentada foi baseada no uso do PSO como um método de otimização de projeto de estruturas *offshore* a fim de facilitar a busca por melhores soluções, evitando uma análise paramétrica manual exaustiva. Nesse contexto, foram apresentados estudos visando a aprimorar e customizar o método de enxame de partículas para aplicação na otimização de *risers*, uma vez que os parâmetros deste método tem se mostrado significativamente dependentes da aplicação considerada. A partir dos resultados obtidos nesses estudos, o método foi implementado na ferramenta *ProgOtim* com os melhores valores e comportamentos da variação para os parâmetros incorporados no código, constituindo uma ferramenta de otimização "amigável" que ajusta automaticamente a maioria dos parâmetros relevantes, customizada para o projeto de *risers*.

Embora o uso de métodos de otimização torne mais fácil o projeto de sistemas *offshore*, evitando uma análise paramétrica manual exaustiva, a análise dinâmica do modelo de elementos finitos ainda deve ser aplicada em cada etapa do algoritmo de otimização. Procurando diminuir o custo computacional desse processo, a segunda metodologia apresentada envolveu estudos sobre a aplicação de modelos substitutos (metamodelos) associados à análise dinâmica de *risers* e linhas de ancoragem, a fim de reduzir o tempo computacional necessário para a obtenção de uma série temporal de tração no topo. Os resultados experimentais mostraram que a rede neural como um modelo exógeno autoregressivo não-linear (NARX) proposta neste trabalho obteve melhores resultados do que a rede neural como um modelo com entradas exógenas, utilizado em pesquisas anteriores (GUARIZE *et al.*, 2007). Os resultados também mostraram que o uso de uma rede neural artificial como um modelo NARX pode fornecer predições de tração no topo com grande precisão, 21 vezes mais rapidamente

do que uma simulação completa de 3h usando o método de elementos finitos, uma vez que somente 500s devem ser calculados a fim de construir os conjuntos de treinamento e validação.

O uso de metamodelos na predição de séries temporais de análises dinâmicas se mostrou uma boa ferramenta para diminuir o custo computacional de procedimentos de otimização na busca por configurações ótimas de sistemas de *risers*. Para a criação de tais metamodelos basta tomar como parâmetro de entrada da rede neural os dados de movimentos da embarcação durante todo o tempo de simulação, reduzindo o tempo de análise necessário para obter os parâmetros de saída (esforços, tensões, etc). Esse procedimento deve ser repetido a cada passo do processo de otimização para cada configuração candidata.

Entretanto, o custo computacional pode ser reduzido ainda mais através da utilização de metamodelos diretamente no processo de otimização. Tal procedimento torna viável o uso de metamodelos em sistemas de ancoragem, onde os movimentos não são dados conhecidos, e sim resultados obtidos de análises de modelos acoplados do casco, linhas de ancoragem e dos *risers*. Sendo assim, a terceira metodologia proposta envolveu estudos sobre a utilização de uma estratégia de otimização de projeto baseada em metamodelagem onde uma rede neural foi treinada a partir de exemplos gerados no processo de otimização a fim de ser capaz de aproximar diretamente a resposta dos sistemas, sem ser necessário o conhecimento do comportamento estrutural inicial dos mesmos. Os resultados experimentais realizados mostraram que a substituição da análise rigorosa pela rede neural na otimização, pode representar uma economia de 54% no tempo computacional necessário para otimizar um modelo de sistema de ancoragem.

Esses resultados são de alta importância prática para a indústria petroleira, uma vez que permitirão a definição de configurações mais eficientes e econômicas de sistemas de ancoragem e de *risers*, levando a projetos mais eficientes para tais sistemas estruturais que são fundamentais na exploração de petróleo *offshore*.

9.2 Trabalhos Futuros

Com o aprimoramento das metodologias aqui apresentadas, pode-se desenvolver uma implementação de Redes Neurais Artificiais no contexto do ambiente computacional SITUA-PROSIM para análise e projeto de sistemas *offshore*. Esse sistema vem sendo desenvolvido no PEC ao longo dos últimos anos, como resultado das pesquisas nesta linha.

No que se refere à estratégia de otimização de projeto baseada em metamodelagem, pode-se estudar o desempenho de uma otimização adaptativa, onde o metamodelo seria ajustado durante o processo de otimização, intercalando etapas de aquisisão de novas amostras de treinamento a partir do modelo de simulação com etapas em que o metamodelo substitui o modelo rigoroso.

Além disso, pode-se investigar a aplicação de outras técnicas de metamodelagem na análise de sistemas offshore.

10 REFERÊNCIAS BIBLIOGRÁFICAS

ABELÉM, A.J.G., 1994, *Redes Neurais Artificiais na Previsão de Séries Temporais*. Dissertação de M.Sc., PUC, Rio de Janeiro.

ABIDO, M., 2002, "Optimal Design of Power System Stabilizers Using Particle Swarm Optimization". *IEEE Trans Energy Convers*, 17(3):406–13.

ABREU, B.T., 2006, *Uma Abordagem Evolutiva para a Geração Automática de Dados de Teste*. Dissertação de M.Sc, Universidade Estadual de Campinas - UNICAMP, Campinas, SP.

ALBRECHT, C.H., 2005, *Algoritmos Evolutivos Aplicados à Síntese e Otimização de Sistemas de Ancoragem*. Tese de D.Sc, COPPE/UFRJ, Rio de Janeiro.

ALMEIDA, C.P., 2007, *Aplicação de Sistemas Imunológicos Artificiais para a Predição da Estrutura de Proteínas*. Dissertação de M.Sc., UTFPR, Curitiba/PR.

AMERICAN BUREAU OF SHIPPING, 1996, "Guide for Building and Classing Floating Production, Storage and Offloading Systems".

ANDERSON, J.A., 1972, "A Simple Neural Network Generating an Interactive Memory". *Mathematical Biosciences*, 14: 197-220. Reprinted in Anderson & Rosenfeld (1988), p. 181-192.

ANDERSON, J.A., 1977, "Neural Models With Cognitive Implications". In: *D. LaBerge & S. J. Samuels (Eds.), Basic Processes in Reading Perception and Comprehension Models*, p. 27-90. Hillsdale, NJ: Erlbaum.

API, 1996, *Recommended Practice for Design and Analysis of Stationkeeping System for Floating Structures, API-RP-2SK*.

API, 1997, *Specification 17J, Specification for Unbonded Flexible Pipes*. American Petroleum Institute, 1st Edition.

API, 2001*, Risk-based Methodologies or Evaluating Petroleum Hidrocarbons Impacts at Oil and Natural Gás E&P Sites*. API Publication Number 4709.

ASTRUP, O.C., NESTEGARD, A., RONOESS, M., SODAHL, N., 2001, "Coupled Analysis Strategies for Deepwater Spar Platforms" – ISOPE.

BÄCK, T., FOGEL, D.B., MICHALEWICZ, Z., 2000a, *Evolutionary Computation 1 Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, UK.

BÄCK, T., FOGEL, D.B., MICHALEWICZ, Z., 2000b, *Evolutionary Computation 2 Advanced Algorithms and Operators*. Institute of Physics Publishing, Bristol, UK.

BÄCK, T., HAMMEL, U., SCHWEFEL, H.P; 1997, "Evolutionary Computation: Comments on the History and Current State" *, IEEE Transaction on Evolutionary Computation*, Vol 1, Nr 1, April 1997, pp 3-17.

BADIRU, A.B., SIEGER, D.B., 1993, *Neural Network as a Simulation Metamodel in Economic Analysis of Risky Projects*, Technical Report, Department of Industrial Engineering, University of Oklahoma.

BAHIENSE, R.A., 2007, *Implementação e Avaliação de uma Metodologia Fortemente Acoplada para Análise de Sistemas Flutuantes Offshore*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro.

BARTHELEMY, J.F.M., HAFTKA, R.T., 1993, "Approximation Concepts for Optimum Structural Design - A Review", *Structural Optimization*, Vol. 5, pp. 129-144.

BARTON, R.R., 1992, "Metamodels For Simulation Input-Output Relations", *Proceedings of the 1992 Winter Simulation Conference*, pp. 289-299.

BARTON, R.R., 1998, "Simulation Metamodels", *Proceedings of the 1998 Winter Simulation Conference*, pp.167-174.

BATTITI, R., 1992, "First and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method". *Neural Comput.*, Vol. 4, p. 141-166.

BAZARGAN, H., BAHAI, H., ARYANA, F., YASSERI, S.F., 2006, "Simulation of the Mean Zero-Up-Crossing Wave Period Using Artificial Neural Networks Trained With Simulated Annealing Algorithm". *Proceedings of OMAE2006, 25th International Conference on Offshore Mechanics and Arctic Engineering*, Hamburg, Germany.

BERNARDO, M.C., BUCK, R., LIU, L., 1992, "Integrated Circuit Design Optimization Using a Sequential Strategy", *IEEE Transactions on Computer-Aided Design*, Vol. 11, n. 3, pp. 361-372.

BEYER, H., SCHWEFEL, H., 2002, "Evolution Strategies - A Comprehensive Introduction". *Natural Computing: an international journal*, 1(1):3–52.

BI, J., BENNETT, K.P., 2003, "Regression Error Characteristic Curves", *Proceedings of the 20th International Conference on Machine Learning*, pp. 43–50.

BLANNING, R.W, 1974, "The Sources and Uses of Sensitivity Information". *Interfaces*. Vol. 4, No. 4, pp. 21-23.

BLANNING, R.W, 1975, "Response to Michel, Kleijnen and Permut". *Interfaces*. Vol. 5, No. 3, pp. 24-25.

BRENT, R.P., 1973, *Algorithms for Minimization Without Derivatives*. Englewood Cliffs, NJ: Prentice-Hall, Chapter 5.

BROOMHEAD, D., LOWE, D., 1988, "Multivariable Functional Interpolation and Adaptive Networks". *Complex Systems*, 2:321-355.

CAMPOS, T.J., 2007, *Hardware Evolutivo Aplicado à Geração Automática de Controladores para Servo-Mecanismos*. Tese de D.Sc., DCA/FEEC/UNICAMP.

CARPENTER, G.A., GROSSBERG, S., 1985, "Category Learning and Adaptive Pattern Recognition, a Neural Network Model". *Proceedings of the Third Army Conference on Applied Mathematics and Computation*, ARO Report 86-1, p. 37-56.

CARVALHO, A.P.L.F., DELBEM, A., SIMÕES, E.V., TELLES, G.P., ROMERO, R.A., 2004, "Computação Bioinspirada". In: *Apostila de minicurso XXIII JAI - Jornada de Atualização em Informática*, Porto Alegre: Sociedade Brasileira de Computação, p. 50.

CASSINI, C.C.S., AGUIRRE, L.A., 1999, *Uma Introdução à Identificação de Sistemas Não Lineares* , Relatório #PPGEE/MACSIN-03/99, Macsin, Belo Horizonte.

CHATTERJEE, A., SIARRY, P., 2006, "Nonlinear Inertia Weight Variation for Dynamic adaptation in Particle Swarm Optimization". *Computers & Operations Research*, Elsevier, 33(3):859–871.

CHENG, B., TITTERINGTON, D.M., 1994, "Neural Networks: A Review from a Statistical Perspective". *Statistical Science*, Vol. 9, n. 1, p. 2-54.

CLOUGH, R.W., PENZIEN, J., 1975, *Dynamic of Structures*. California, McGraw-Hill.

COELHO, L.S., 2003, "Fundamentos, Potencialidades e Aplicações de Algoritmos Evolutivos". In *E. X. L. de Andrade, R. Sampaio, and G. N. Silva, editors, Notas em Matemática Aplicada*. SBMAC.

COLLIAT, J.L., 2002, *Total Fina ELF. Anchors for Deepwater and Ultradeepwater Moorings*. OTC 14.306 – 2002. Houston – USA.

CRESSIE, N.A.C., 1993, *Statistics for Spatial Data, Revised*, John Wiley & Sons, New York.

CUNHA, A.P., 2001, *Redes Neurais em Processos Siderúrgicos: Analisador Virtual de Propriedades Metalúrgicas do Síter e Modelo Modelo de Predição de Qualidade do Aço*. Tese de D.Sc., UNICAMP, Campinas, SP.

CYBENKO, G., 1989, "Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control", *Signals and Systems*, Vol. 2, p. 303-314.

DARPA, 1988, *DARPA Neural Network Study. Final Report*, Cambridge, MA: Massachusetts Institute of Technology, Lincoln Laboratory.

DASGUPTA, D., 1998, "An Overview of Artificial Immune Systems and Their Applications". *Artificial Immune Systems and Their Applications.* Springer-Verlag, Berlin, Germany.

DE CASTRO, L.N., 2001, *Engenharia Inmunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais*. Tese de D.Sc, Universidade Estadual de Campinas - Faculdade de Engenharia Elétrica e de Computação.

DE CASTRO, L.N., VON ZUBEN, F.J., 2000, "The Clonal Selection Algorithm With Engineering Applications". In: *GECCO. Workshop on Artificial Immune Systems and Their Applications*. Las Vegas, Nevada, USA, p. 36–37.

DE JONG, K. A., 2006, *Evolutionary computation: A Unified Approach*. Cambridge, MA: MIT Press.

de LIMA, B.S.L.P., JACOB, B.P., EBECKEN, N.F.F., 2005, "A Hybrid Fuzzy/Genetic Algorithm For The Design Of *Offshore* Oil Production *Risers*". *International Journal For Numerical Methods In Engineering*. Int. J. Numer. Meth. Engng 2005; 64:1459–1482.

DELBEM, A.C.B., CARVALHO, A.C.P.L.F., BRETAS, N.G., 2005, "Main Chain Representation For Evolutionary Algorithm Applied To Distribution System Reconfiguration". *IEEE Transactions on Power Systems*, 20(1), p. 425–436.

DENNIS, J.E., TORCZON, V., 1996, "Managing Approximation Models in Optimization," In: Alexandrov, N. and Hussaini, M.Y. (Editors), *Multidisciplinary Design Optimization: State of the Art, Society for Industrial and Applied Mathematics*, Philadelphia.

EBERHARDT, R.C., SHI, Y., 1999,"Empirical Study of Particle Swarm Optimization". *IEEE proc. Conference on Evolutionary Computation*, pp.1945 – 1949.

EBERHARDT, R.C., SHI, Y., 2000, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization". *IEEE International Conference on Evolutionary Computation*, San Diego, California, pp. 84 – 88.

EBERHARDT, R.C., SHI, Y., 2004, "Guest Editorial: Special Number on Particle Swarm Optimization". *Transaction on Evolutionary Computation, IEEE*, 8(3).

ELTAHER, A., RAJAPAKSA, Y., CHANG, K., 2003, *Industry Trends for Design of Anchoring Systems for Deepwater Offshore Structures*, OTC 15265, Offshore Technology Conference.

FAUSETT, L.V., 1994, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice Hall.

FINE, T.L., 1999, *Feedforward Neural Network Methodology*. Springer.

FISHWICK, P.A., 1989, "Neural Network Models In Simulation: A Comparison With Traditional Modeling Approaches", *Proceedings of the 1989 Winter Simulation Conference*, 702-710.

FONSECA, L.G., 2009, *Algoritmos Genéticos Assistidos por Metamodelos Baseados em Similaridade*. Tese de D.Sc., LNCC/MCT, Petrópolis, RJ.

FONTOVA, M.I.V., 2003, *Solução de Problemas de Otimização Linear por Redes Neurais Associadas a Métodos de Pontos Interiores*. Tese de D.Sc., UNICAMP, Campinas, SP, Brasil.

FRANKE, R., 1982, "Scattered Data Interpolation: Tests of Some Methods". *Mathematics of Computation* 38, pp.181-200.

FRIEDMAN, J.H., 1991, "Multivariate Adaptive Regression Splines". *The Annals of Statistics*, 19(1), pp. 1-141.

GABRIEL, P.H.R., 2010, *Algoritmos Evolutivos e Modelos Simplificados de Proteínas para Predição de Estruturas Terciárias*. Dissertação de M.Sc., ICMC-USP - São Carlos.

GABRIEL, P.H.R., DELBEM, A.C.B., 2008, *Fundamentos de Algoritmos Evolutivos*. Notas Didáticas do ICMC-USP, ISSN 0103-2585, Nº. 75, 35 pg.

GARZA-RIOS, L.O., BERNITSAS, M.M., NISHIMOTO, K., MASETTI, I.Q., 1997, "Preliminary Design of a DICAS Mooring Systemm for the Brazilian Campos Basin". *OMAE*, Vol. I-A, *Offshore Technology*, *ASME*.

GASPAR-CUNHA, A., VIEIRA, A., 2009, "A Multi-Objective Evolutionary Algorithm Using Neural Networks To Approximate Fitness Evaluations". *International Journal of Computers, Systems and Signals*, Vol. 9, No. 9.

GIUNTA, A., WATSON, L.T., KOEHLER, J., 1998, "A Comparison of ApproximationModeling Techniques: Polynomial Versus Interpolating Models", *7th AIAA/USAF/NASA/ISS Symposium on Multidisciplinary Analysis & Optimization*, St. Louis, MO, AIAA, Vol. 1, pp. 392-404. AIAA-98-4758.

GOLDBERG, D.E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

GOMES, M.V.C., 2007, *Otimização Seqüencial por Aproximações – Uma Aplicação Em Tempo Real para o Refino de Petróleo*. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro.

GONÇALVES, R.C.F, COSTA, L.C.S, 2002, "Inspeção em Linhas de Ancoragem de Unidades de Produção", *XXI Congresso Nacional de Ensaios Não Destrutivos*, Salvador.

GREFENSTETTE, J.J., FITZPATRICK, J.M., 1985, "Genetic Search With Approximate Function Evaluations". *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pp. 112-120.

GU, L., 2001, "A Comparison of Polynomial Based Regression Models in Vehicle Safety Analysis". In: Diaz, A. (Ed.), 2001 ASME *Design Engineering Technical Conferences - Design Automation Conference*, ASME, Pittsburgh, PA, September 9-12, DAC-21063.

GUARIZE, R., 2004, *Uso de Redes Neurais Na Análise de Resposta Dinâmica de Estruturas*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

GUARIZE, R., MATOS, N.A.F., SAGRILO, L.V.S., LIMA, E.C.P., 2007, "Neural Networks in the Dynamic Response Analysis of Slender Marine Structures". *Applied Ocean Research* 29, 191–198.

GUPTA, M.M., JIN, L., HOMMA, N., 2003, *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*. John Wiley & Sons, Inc., Hoboken, New Jersey.

HAGAN, M.T., MENHAJ, M., 1994, "Training Feed-forward Networks With the Marquardt Algorithm", *IEEE Transactions on Neural Networks*, vol. 5, no. 6, p. 989–993.

HARDY, R.L., 1971, "Multiquadratic Equations of Topography and Other Irregular Surfaces", *J. Geophys. Res.*, Vol. 76, pp. 1905-1915.

HARRALD, P. G., KAMSTRA, M., 1997, "Evolving Artificial Neural Networks to Combine Financial Forescasts", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, p. 40–52.

HASSAN, R., COHANIM, B., DE WECK, O., VENTER, G., 2005, "A Comparison of Particle Swarm Optimization and the Genetic Algorithm". *In 1st AIAA Multidisciplinary Design Optimization Specialist Conference*, Austin, TX, No. AIAA-2005-1897.

HAYKIN, S., 1999, *Neural Networks - A Comprehensive Foundation*. 2nd edition. Prentice Hall, New Jersey.

HE, S., WU, Q.H., WEN, J.Y, SAUNDERS, J.R., PATON, R.C., 2004, "A Particle Swarm Optimizer With Passive Congregation". *Elsevier BioSystems*, Vol. 78, pp. 135–147.

HEBB, D.O., 1949, The *Organization of Behavior*. New York: John Wiley & Sons. Introduction and Chapter 4 reprinted in Anderson & Rosenfeld (1988), pp.45-56.

HESS, D.E., FALLER, W.E., RODDY Jr., R.F., PENCE, A.M., FU, T.C., 2006, "Feedforward Neural Networks Applied To Problems In Ocean Engineering". *Proceedings of OMAE2006, 25th International Conference on Offshore Mechanics and Arctic Engineering*, Hamburg, Germany.

HEURTIER, J.M., BUHAN, P. Le, FONTAINE, E., CUNFF, C. Le, BIOLLEY, F., BERHAULT, C., 2001, "Coupled Dynamic Response of Moored FPSO with Risers". *ISOPE*.

HOLLAND, J.H., 1975, *Adaptation in natural and artificial systems*. University of Michigan Press.

HOPFIELD, J.J., 1982, "Neural Networks and Physical System With Emergent Collective Computacional Abilities". *Proceedings of the National Academy of Sciences of the U.S.A.*, Vol. 79, p. 2554-2558.

HU, X., EBERHARDT, R.C., SHI, Y., 2003, "Engineering Optimization With Particle Swarm". *In IEEE Swarm Intelligence Symposium* (SIS 2003), pp. 53 – 57.

HUSSAIN, M.F., JOSHI, S., BARTON, R.R, 2002, "Metamodeling: Radial Basis Functions Versus Polynomials", *European Journal of Operational Research 138*, 142-154.

IYODA, E.M., 2000, *Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas*. Dissertação de M.Sc., UNICAMP, Campinas, SP, Brasil.

IYODA, E.M., DE CASTRO, L.N., GOMIDE, F., VON ZUBEN, F.J., 1999, "Evolutionary Design of Neurofuzzy Networks for Pattern Classification". *Proc. of the 1999 Congress on Evolutionary Computation*, Vol. 2, p. 1237-1244.

JACOB, B.P, 2005. *Programa PROSIM: Simulação Numérica do Comportamento de Unidades Flutuantes Ancoradas, Versão 2.7b– Manual de Entrada de Dados*, COPPE/UFRJ, Programa de Engenharia Civil, Rio de Janeiro.

JACOB, B.P., LIMA, B.S.L.P., REYES, M.C.T., 1998, *Estudos paramétricos em configurações alternativas de SCRs conectadas a FPSO's*. LAMCSO/COPPE/ Universidade Federal do Rio de Janeiro - Petrobras ET-150822, Outubro.

JACOB, B.P., LIMA, B.S.L.P., REYES, M.C.T., 2001, *Estudos paramétricos da configuração Lazy-Wave para SCR em FPSO's*. LAMCSO/COPPE/ Universidade Federal do Rio de Janeiro - Petrobras PEC-361, Janeiro.

JACOB, B.P., LIMA, B.S.L.P., REYES, M.C.T., TORRES, A.L.F.L., MOURELLE, M.M., SILVA, R.M.C., 1999, "Alternative Configurations for Steel Catenary Risers for Turret-Moored FPSO's". *Proceedings of the 9th International Offshore and Polar Engineering Conference*, Brest- France; II: 234-239.

JACOB, B.P., MASETTI, I.Q., 1997, *Prosim – Coupled Numerical Simulation of the Behavior Of Moored Semisubmersible Units*. COPPETEC-Petrobras Internal Report, Rio de Janeiro.

JIN, R., CHEN, W., SIMPSON, T. W., 2000, "Comparative Studies of Metamodeling Techniques Under Multiple Modeling Criteria". *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Multidisciplinary Analysis & Optimization Symposium*, AIAA 2000-4801, Long Beach, CA.

JOHANSSON, E.M., DOWLA, F.U., GOODMAN, D.M., 1990, *Backpropagation Learning for Multi-Layer Feed-Forward Neural Networks Using the Conjugate Gradient Method*. Lawrence Livermore National Laboratory, Berkeley, CA, Preprint UCRL-JC-104850.

JUVINAO CARBONO, A. J., 2005, "Otimização da disposição de linhas de ancoragem utilizando algoritmos genéticos", Dissertação de M.Sc, PUC-Rio.

KANNAN, S., SLOCHANAL S.M.R., SUBBARAJ P., NARAYANA P.P., 2004, "Application of Particle Swarm Optimization Technique and its Variants to Generation Expansion Planning Problem". *Electric Power Systems Research*, Vol. 70, pp. 203 – 210 , Elsevier.

KENNEDY, J., EBERHARDT, R., 1995a, "A New Optimizer Using Particle Swarm Theory". *Sixth International Symposium on Micro Machine and Human Science, IEEE*, pp. 39 – 43.

KENNEDY, J., EBERHARDT, R., 1995b, "Particle Swarm Optimizarion". *Proc. IEEE Conference on Neural Networks*, pp. 1942 – 1948.

KENNEDY, J., EBERHARDT, R.C., SHI, Y., 2001, "Swarm Intelligence". *The Morgan Kaufmann Series in Evolutionary Computation*, Academic Press, San Francisco.

KHOSLA, R., DILLON, T., 1997, *Engineering Intelligent Hybrid Multi-Agent Systems*. Kluwer Academic Publishers: Boston, USA.

KIM, M.H., WARD, E.G., HARING, R., 2001, "Comparison of Numerical Models for the Capability of Hull/Mooring/Riser Coupled Dynamic Analysis for Spars and TLPs in Deep and Ultra-Deep Waters", *ISOPE*.

KINGRE, H.J., 2004, *Bezier Curve for Metamodeling of Simulation Output*. Master's Thesis. Louisiana State University.

KITAMURA, M., DJENOD, K., HAMADA, K., 2002, "Neural Network Based on Finite Element Analysis and its Application to Structural Optimization of Container Ship", *Journal of the Society of Naval Architects of Japan*,192,661-668.

KLEIJNEN, J.P.C., 1975, "A Comment on Blanning's "Metamodel for Sensitivity Analysis: The Regression Metamodel in Simulation"". *Interfaces*. Vol. 5, No. 3, pp. 21-23

KLEIJNEN, J.P.C., SARGENT, R.G., 2000, "A Methodology for Fitting and Validating Metamodels in Simulation". *European Journal of Operational Research*, Vol. 120, pp. 14-29.

KOHONEN, T., 1972, "Correlation Matrix Memories". *IEEE Transactions on Computers, C-21:353-359. Reprinted: Anderson & Rosenfeld (1988)*, p.174-180.

KOHONEN, T., 1977, *Associative Memory: A System-Theoretical Approach*. Springer-Verlag.

KOHONEN, T., 1982, "Self-organised Formation of Topologically Correct Feature Maps". *Biological Cybernetics*, Vol. 43, p. 59–69. *Reprinted in Anderson & Rosenfeld (1988)*, pp. 511-521.

KÓVACS, Z.L., 2002*, Redes Neurais Artificiais*. São Paulo, Editora Livraria da Física.

KRISHNAKUMAR, K., 1989, "Micro-Genetice Algorithms for Stationary and Non-Stationary Function Optimization". *SPIE: Inteligent Control and Adaptative Systems*, v. 1196, p. 289–296.

LE CUN, Y., 1986, "Learning Processes in an Asymmetric Threshold Network". In E. Bienenstock, F. Fogelman-Souli, & G. Weisbuch, eds. *Disordered Systems and Biological Organization*. NATO ASI Series, F20, Berlin: Springer-Verlag.

LEMMON, M., SZYMANSKI, P.T., 1994, "Interior Point Implementations of Alternating Minimization Training". *Advances in Neural Information Processing Systems*, Vol. 7, pp. 574-582.

LEVENBERG, K., 1944, "A Method for the Solution of Certain Non-linear Problems in Least Squares". *Quarterly of Applied Mathematics*, vol. 2, no. 2, p. 164–168.

LINKSER, R., 1988. "Self-Organization in Perceptual Network". *Computer*, p.105-117.

LIPPMANN, R.P., 1987, "An Introduction to Computing with Neural Nets". *IEEE ASSP Magazine*, 4:4-22.

MAHDI, N., MANSOOREH, M., 1999, "The Development of a Methodology for the Use of Neural Networks and Simulation Modeling In System Design", *Proceedings of the 1999 Winter Simulation Conference*, pp.537-542.

MAHFOUZ, A.B., 2006, "Predicting The Capability-Polar-Plots For Dynamic Positioning Systems For *Offshore* Platforms Using Artificial Neural Networks". *Ocean Engineering*, doi:10.1016/j.oceaneng.2006.08.006.

MARQUARDT, D.W., 1963, "An Algorithm for the Least-Squares Estimation of Nonlinear Parameters". *SIAM Journal of Applied Mathematics*, vol. 11, no. 2, p. 431–441.

MARTINS, M.A.L., 2008, *Avaliação das Metodologias de Projeto de Risers Rígidos*. Monografia de Graduação. UFAL, Alagoas.

MASETTI, I.Q.; 1997, *Análise Dinâmica de Navios Ancorados com Complacência Diferenciada*. Tese de D.Sc., COPPE/UFRJ.

MATOS, N.A.F., 2005, *Predição de Séries Temporais para Análise Dinâmica de Estruturas Offshore*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro.

MAZAHERI, S., DOWNIE, M.J., 2004, "Response-Based Method For Determining The Extreme Behaviour Of Floating *Offshore* Platforms". *Ocean Engineering*, doi:10.1016/j.oceaneng.2004.08.004

MCCULLOCH, W.S., PITTS W., 1943, "A Logical Calculus of the Ideas Immanent in Nervous Activities". *Bulletin of Mathematical Biophysics*, Vol. 5, p. 115-133.

MEDEIROS, C.J.Jr., 2002, *Low Cost Anchor System For Flexible Risers in Deep Waters*. OTC 14.309 – 2002. Houston – USA.

MEDEIROS, J.A.C.C., 2005, *Enxame de Partículas como Ferramenta de Otimização em Problemas Complexos de Engenharia Nuclear*. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro.

MENDONÇA, C.E.L.R., 2004, *Um Sistema Computacional para Otimização Através de Algoritmos Genéticos e Redes Neurais*. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro.

MICHALEWICZ, Z., 1996, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd Edition, Springer Verlag, Berlin Heidelberg.

MICHEL, A.J., PERMUT, S.E., 1975, "A Comment on Blanning's "The Sources and Uses of Sensitivity Information"". *Interfaces*, Vol. 5, No. 3, pp. 19-20.

MINSKY, M.L., PAPERT, S.A., 1969, *Perceptrons*. Cambridge: MIT Press.

MITCHELL, T.M., 1997, *Machine Learning*, Mcgraw-Hill.

MOLLER, M.F., 1993, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning". *Neural Network*, Vol. 6, No. 4, p.525-534.

MONTEIRO, B.F., 2008, *Aplicação do Método do Enxame de Partículas na Otimização de Sistemas de Ancoragem de Unidades Flutuantes para Explotação de Petróleo Offshore*. Dissertação de M.Sc., COPPE/UFRJ.

MYERS, R.H. AND MONTGOMERY, D., 1995, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley and Sons, Inc., Toronto.

MYERS, R.H., MONTGOMERY, D.C., 2002, *Response Surface Methodology-Process and Product Optimization Using Designed Experiment*, New York: John Wiley & Sons.

NASEREDDIN, M., MOLLAGHASEMI, M., 2005, "Exploring the Choice of Experimental Design Used to Create the Training Set for a Reverse Neural Network Simulation Metamodel In System Design". *Asia J. Inform. Technol.*, 4: 1102-1109.

ORMBERG, H., FYLLING, I. J., LARSEN, K., 1997, "Coupled Analysis of Vessel Motions and Mooring and Riser System Dynamics". *OMAE*, v I-A, *Offshore Technology*, pp. 91-100, 1997.

OSIO, I.G.; AMON, C.H., 1996, "An Engineering Design Methodology with Multistage Bayesian Surrogates and Optimal Sampling". *Research in Engineering Design*, 8(4), 189-206.

OURIQUE, C.O., BISCAIA JR., E.C., PINTO, J.C., 2002, "The Use of Particle Swarm Optimization for Dynamical Analysis in Chemical Processes". *Computers and Chemical Engineering*, Vol. 26, pp. 1783 – 1793.

PADGETT, M.L., ROPPEL,T.A., 1992, "Neural Networks and Simulation: Modelling For Applications", *Simulation 58*, 295-305.

PARKER, D., 1985, *Learning Logic*. Technical Report TR-87, Cambridge, MA: Center for Computational Research in Economics and Management Science, MIT.

PARSOPOULOS, K.E., VRAHATIS, M.N., 2002, "Recent Approaches To Global Optimization Problems Through Particle Swarm Optimization", *Natural Computing* 1:235–306.

PARSOPOULOS, K.E., VRAHATIS, M.N., 2004, "On the Computation of All Global Minimizers Through Particle Swarm Optimization". *Transaction on Evolutionary Computation, IEEE*, Vol. 8, No. 3, pp. 211–224.

PENG, J., CHEN, Y., EBERHART R., 2000, "Battery Pack State of Charge Estimator Design Using Computational Intelligence Approaches". In: *Fifteenth Annual Battery Conference on Applications and Advances*, pp. 173–177.

PETROBRAS, 1999, *Metocean Data – Technical Specification,* ET-3000.00-1000-941-PPC-001.

PIERREVAL, H., HUNTSINGER, R.C., 1992, "An investigation on neural network capabilities as simulation metamodels," *Proceedings of the 1992 Summer Computer Simulation Conference*, 413-417.

PIERREVAL, H., 1992, "Training a Neural Network by Simulation for Dispatching Problems", *Proceedings of the Third Rensselaer International Conference on Computer Integrated Engineering*, 332-336.

PINA, A.A., ALBRECHT, C.H., LIMA, B.S.L.P., JACOB, B.P., 2010a, "Tailoring the Particle Swarm Optimization Algorithm for the Design of *Offshore* Oil Production *Risers*". *Optmization and Engineering*, v. Online, p. 1.

PINA, A.A., LIMA, B.S.L.P., ALBRECHT, C.H, JACOB, B.P., 2008, "Parameter Selection and Convergence Analysis of the PSO Algorithm Applied to the Design of Risers". *Proc. of EngOpt 2008 - International Conference on Engineering Optimization*, Rio de Janeiro.

PINA, A.A., PINA, A.C., ALBRECHT, C.H., de LIMA, B.S.L.P, JACOB, B.P., 2010b, "Applying Computational Intelligence in the Design of Moored Floating Systems for Offshore Oil Production". *Proceedings of the 2nd International Conference on Engineering Optimization EngOpt 2010*, Lisboa.

PINA, A.C., ZAVERUCHA, G., 2008, "Applying REC Analysis to Ensembles of Particle Filters", *Neural Computing & Applications - Special Issue on Temporal Data Analysis*, Springer London, Inglaterra, doi: 10.1007/s00521-008-0199-x.

POLI R, KENNEDY J AND BLACKWELL T , 2007, "Particle Swarm Optimization: An Overview", *Swarm Intell* 1:33–57, Doi:10.1007/s11721-007-0002-0.

PROVOST, F., FAWCETT, T., 1997, "Analysis and Visualization of Classifier Performance: Comparison Under Imprecise Class and Cost Distributions", *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Newport Beach, CA, 43–48, 1997.

RANGANATHAN, A., 2004, *The Levenberg-Marquardt Algorithm*.

RATNAWEERA A., HALGAMUGE S.K., 2004, "Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients". *Transactions On Evolutionary Computation, IEEE*, Vol. 8, No. 3, June .

RECHENBERG, I., 1965, *Cybernetc Solution Path of an Experimental Problem*. Relatório Técnico 1122, Royal Aircraft Establishment, Franborough, UK.

ROCHA, D.M, 2007, *Redes Neurais para Modelagem de Sistemas Estruturais Offshore Dinâmicos Não-Lineares com Histerese*. Tese de D.Sc., COPPE/UFRJ.

RODRIGUES, G.J.O, 2004, *Ferramentas Computacionais para Otimização e Síntese de Sistemas Híbridos de Risers Baseados No Conceito de Bóia de Subsuperfície*. Tese de D.Sc., COPPE/UFRJ.

ROSENBLATT, F., 1958, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". *Psychological Review*, Vol. 65, p. 386-408.

ROSSI, R.R., MESSINA, L.C.P., AMARANTE, L.M.N., KAISER, A.M., AUGUSTO, O.B., SILVA, J.R., 2001, "MODU's Mooring Systems for Campos Basin Deep Water Water Fields". *OMAE*.

RUMELHART, D.E, MCCLELLAND, J.L., 1986, "Parallel Distributed Processing", *Explorations in the Microstructure of Cognition*. Vol 1: Foundations. Cambridge, MA: MIT Press.

SANTOS, C.M.P.M., 2000, *Otimização de Um Sistema de Ancoragem de um FPSO*, Exame de Qualificação ao Doutorado, COPPE/UFRJ, Rio de Janeiro.

SANTOS, I.R., SANTOS, P.R., 2007, "Simulation Metamodels For Modeling Ouput Distribution Parameters". *In Proceedings of the 2007 Winter Simulation Conference*, ed. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J.D. Tew, and R.R. Barton.

SCHONLAU, M.S., WELCH, W.J., JONES, D.R., 1998, "Global Versus Local Search in Constrained Optimization of Computer Models". In: Flournoy, N., Rosenberger, W.F. and Wong, W.K. (Editors), *New Development and Applications in Experimental Design*, Institute of Mathematical Statistics, pp. 11-25.

SCHWEFEL, H.P., 1995, *Evolution and Optimun Seeking,* John Wiley & Sons, Inc., New York.

SENRA, S.F., 2004, *Metodologias de Análise e Projeto Integrado de Sistemas Flutuantes para Explotação de Petróleo Offshore*, Tese de D.Sc., COPPE/UFRJ.

SETTLES, M., RYLANDER, B., 2002, "Neural Network Learning Using Particle Swarm Optimizers". *Advances in Information Science and Soft Computing*, pp. 224-226.

SHAN, S. AND WANG, G.G., 2005a, "An Efficient Pareto Set Identification Approach for Multi-objective Optimization on Black-box Functions," *Transactions of the ASME, Journal of Mechanical Design*, 127, 866-874.

SHAN, S., WANG, G. G., 2005b, "Failure Surface Frontier for Reliability Assessment on Expensive Performance Function," *Transactions of ASME, Journal of Mechanical Design*.

SHEWCHUK, J.R., 1994, *An Introduction to the Conjugate Gradient Method Without te Agonizing Pain*. Pittsburgh, School of Computer Science Carnegie Mellon University.

SHI, Y., EBERHARDT, R.C., 1998, "A Modified Particle Swarm Optimizer". *IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, pp. 1945–1950.

SIDDAL, J.N., 1982, *Optimal Engineering Design: Principles and Applications*, Marcel Drekker, inc., New York.

SIMPSON, T.W., MAUERY, T.M., KORTE, J.J., MISTREE, F., 1998, "Comparison of Response Surface and Kriging Models for Multidisciplinary Design Optimization", *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis & Optimization*, St. Louis, MO, AIAA, Vol. 1, pp. 381-391. AIAA-98-4755, September 2-4.

SMAGT, P., 1994, "Minimisation Methods for Training Feedforward Neural Networks", *Neural Networks*, Vol. 7, No. 1, pp. 1-11.

SOBIESZCZANSKI-SOBIESKI, J., HAFTKA, R.T., 1997, "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments", *Structural Optimization*, 14, pp. 1-23.

SONG, C.Y., LEE, J., CHOUNG, J.M., 2010, "Reliability-Based Design Optimization of an FPSO *Riser* Support Using Moving Least Squares Response Surface Meta-Models". *Ocean Engineering*, in Press.

SONG, L., LI, X., GARCIA-DIAZ, A., 2008, "Multi-Echelon Supply Chain Simulation Using Metamodel". *Winter Simulation Conference (WSC)*.

SPECHT, D.F., 1988, "Probabilistic Neural Networks for Classification, Mapping, or Associative Memory". IEEE *International Conference on Neural Networks*, San Diego, CA, I:525-532.

SUTTON, R.S., BARTO, A.G., 1998, *Reinforcement Learning: An Introduction (Adaptative Computation and Machine Learning)*. MIT Press.

TIMMIS, J., 2000, *Artificial Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory*. Tese de D.Sc., University of Whales, Department of Computer Science, Aberystwyth, Ceredigion, Wales.

TOWSEY, M., ALPSAN, D., SZTRIHA, L., 1995, "Training a Neural Network with Conjugate Gradient Methods". *IEEE*, p. 373-378.

TRAFALIS, T.B., COUELLAN, N.P., 1994, "Neural Network Training via a Primal-Dual Interior Point Method for Linear Programming". *WCNN—World Congress on Neural Networks*, vol. 2, pp. 798-803 .

TRELEA, I.C., 2003, "The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection". *Information Processing Letters*, 85:317–325.

TURING, A.M., 1936, "On Computable Numbers: With an Application to the Entscheidungsproblem". *Proceedings of the Mathematical Society*, Series 2, Vol. 42, p. 230-265.

VAPNIK, V., 1998, *Statistical Learning Theory*, Wiley, New York.

VEELENTURF, L.P.J., 1995, *Analysis and Applications of Artificial Neural Netwoks*. Prentice Hall.

VELAZCO, M.I., LYRA, C., 2002, "Optimization with Neural Networks Trained by Evolutionary Algorithms". *IJNN—International Joint Conference on Neural Networks*, Honolulu, Hawaii/EUA, pp. 1516-1521.

VIEIRA, I.N., 2009, *Algoritmos Bio-Inspirados Aplicados à Otimização de Risers Rígidos Em Catenária*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro.

VIEIRA, I.N., LIMA, B.S.L.P., JACOB, B.P., 2008a, "Optimization of Steel Catenary Risers for Offshore Oil Production Using Artificial Immune System". *In: International Conference on Artificial Immune Systems, Lecture Notes in Computer Science*. Berlin: Springer-Verlag, v. 5132. pp. 254 – 265, Phuket, Thailand.

VIEIRA, I.N., SILVA, A.J.M., LIMA, B.S.L.P, JACOB, B.P., ALBRECHT, C.H., 2008b, "A Comparative Study Applied to Risers Optimization Using Bio-Inspired Algorithms". *In: Proceedings of XXIX CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering*. Maceió, Brasil.

VIEIRA, L.T., 2008, *Otimização de Sistemas de Risers para Explotação de Petróleo Offshore Através de Algoritmos Genéticos Paralelos*. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro.

VIEIRA, L.T., LIMA, B.S.L.P., EVSUKOFF, A.G., JACOB, B.P., 2003, "Application of Genetic Algorithms to the Synthesis of Riser Configurations". *In: Proceedings of the 22th International Conference on Offshore Mechanics and Arctic Engineering*, paper OMAE2003-37231 pp. 1-6, Cancun, Mexico.

VON NEUMANN, J., 1958, *The Computer and the Brain*. New Haven: Yale University Press. p. 66-82. Reprinted in Anderson & Rosenfeld (1988), p. 83-89.

VON ZUBEN, F.J., 2000, "Computação Evolutiva: Uma Abordagem Pragmática". *In: Anais da I Jornada de Estudos em Computação de Piracicaba e Região (1a. JECOMP)*, Piracicaba, SP, p. 25–45.

WANG, G.G., DONG, Z., AITCHISON, P., 2001, "Adaptive Response Surface Method - A Global Optimization Scheme for Computation-intensive Design Problems", *Journal of Engineering Optimization*, 33(6), 707-734.

WANG, G.G., S. SHAN, S., 2007, "Review of Metamodeling Techniques in Support of Engineering Design Optimization". *ASME Transactions, Journal of Mechanical Design* 129(4), 370–380.

WANG, L., SHAN, S., WANG, G.G., 2004, "Mode-Pursuing Sampling Method for Global Optimization on Expensive Black-box Functions", *Journal of Engineering Optimization*, 36(4), 419-438.

WARNER, B., MISRA, M., 1996, "Understanding Neural Networks as Statistical Tools". *The American Statistician*, Vol. 50, n. 4, p. 284-293.

WASSERMAN, P.D., 1989, *Neural Computing – Theory and Practice*. Van Nostran Reinhold.

WERBOS, P., 1974, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph.D. Thesis, Cambridge, MA: Harvard U. Committee on Applied Mathematics.

WICHERS, J.E.W., DEVLIN, P.V., 2001, "Effect of Coupling of Mooring Lines and Risers on the Design Values for a Turret Moored FPSO in Deep Water of the Gulf of Mexico", *ISOPE*.

WIDROW, B., HOFF, M.E., 1960, "Adaptative Switching Circuits". *In IRE WESCON Convention Record*, p. 96–104.

WIDROW, B., LEHR, M.A., 1990, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation". *Proceedings of the IEEE*, 78(9):1415-1442.

WIDROW, B., RUMELHART, D.E., LEHR, M.A., 1994, "Neural networks: Applications in Industry, Business and Science", *Communications of the ACM*, Vol. 37, no. 3, p. 93-105.

WILLIAMS, R.J., PENG, J., 1990, "An efficient Gradient-Based Algorithm for On-line Training of Recurrent Network Trajectories". *Neural Computation*, Vol. 2, p. 490-501.

WINANDY, C.E., BORGES FILHO, E., BENTO, L.V., 2007, *Algoritmos para Aprendizagem Supervisionada*. CT215 – Inteligência Artificial, Seminários.

ZHU, Q.M., 2003, "A Back Propagation Algorithm to Estimate the Parameters of Nonlinear Dynamic Rational Models". *Applied Mathematical Modelling* v.27 pp 169-187.