UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE ENGENHARIA MECÂNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DISSERTAÇÃO DE MESTRADO

Detecção de danos em estruturas por meio de técnicas de redes neurais artificiais e algoritmos genéticos

Autor: **Patricia da Silva Lopes**
Orientador: **Prof. Dr. Ariosto Bretanha Jorge**
Co-orientador: **Prof. Dr. Sebastião Simões da Cunha Jr.**

Itajubá, Março de 2007

UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE ENGENHARIA MECÂNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DISSERTAÇÃO DE MESTRADO

Detecção de danos em estruturas por meio de técnicas de redes neurais artificiais e algoritmos genéticos

Autor: **Patricia da Silva Lopes**
Orientador: **Prof. Dr. Ariosto Bretanha Jorge**
Co-orientador: **Prof. Dr. Sebastião Simões da Cunha Jr.**

Curso: **Mestrado em Engenharia Mecânica**
Área de Concentração: **Projeto e Fabricação**

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Mecânica como parte dos requisitos para obtenção do Título de Mestre em Engenharia Mecânica.

Itajubá, Março de 2007
M.G. – Brasil

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Cristiane N. C. Carpinteiro- CRB_6/1702

L864d

Lopes, Patricia da Silva
Detecção de danos em estruturas por meio de técnicas de redes neurais artificiais e algoritmos genéticos / por Patricia da Silva Lopes. -- Itajubá (MG) : [s.n.], 2007.
106 p. : il.

Orientador : Prof. Dr. Ariosto Bretanha Jorge
Co-Orientador : Prof. Dr. Sebastião Simões da Cunha Jr.
Dissertação (Mestrado) – Universidade Federal de Itajubá

1. Detecção de danos. 2. Redes neurais artificiais. 3. Otimização. 4. Algoritmos genéticos. I. Jorge, Ariosto Bretanha, orient. II. Cunha Jr., Sebastião Simões da, co-orient. III. Universidade Federal de Itajubá. IV. Título.

CDU 004.032.26 (043.3)

UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE ENGENHARIA MECÂNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DISSERTAÇÃO DE MESTRADO

Detecção de danos em estruturas por meio de técnicas de redes neurais artificiais e algoritmos genéticos

Autor: **Patricia da Silva Lopes**
Orientador: **Prof. Dr. Ariosto Bretanha Jorge**
Co-orientador: **Prof. Dr. Sebastião Simões da Cunha Jr.**

Composição da Banca Examinadora:

Prof. Dr. Eder Lima de Albuquerque – FEM/UNICAMP

Prof. Dr. Márcio Tadeu de Almeida - IEM/UNIFEI

Prof. Dr. Ariosto Bretanha Jorge (Orientador) - IEM/UNIFEI

Prof. Dr. Sebastião Simões da Cunha Jr. (Co-orientador) - IEM/UNIFEI

Agradecimentos

Agradeço primeiramente a Deus, pelo dom da vida e pela minha capacidade de pensar.

Aos meus pais Pedro e Ester, pelo incentivo e contribuição no meu sucesso, e sem os quais eu não chegaria até aqui.

Ao meu orientador, prof. Ariosto Bretanha Jorge, por ter acreditado na minha capacidade e aberto novas possibilidades na minha carreira profissional.

Ao prof. Sebastião Simões da Cunha Jr., meu co-orientador, pelo apoio fundamental para a execução deste trabalho.

Aos demais professores do Instituto de Engenharia Mecânica da Unifei, pelo apoio e pelos conhecimentos transmitidos.

Aos funcionários desta instituição, pelo auxílio prestado inúmeras vezes.

Aos pesquisadores do Grupo de Mecânica Computacional, pelo companheirismo, incentivo e apoio.

Ao CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico, pelo apoio financeiro.

“Sem a arte da dúvida,
a ciência não tem como sobreviver e
expandir a sua produção de conhecimento.”
Augusto Cury

Resumo

LOPES, P. S. (2007), *Detecção de danos em estruturas por meio de técnicas de redes neurais artificiais e de algoritmos genéticos*, Itajubá, 106p. Dissertação (Mestrado em Projeto e Fabricação) - Instituto de Engenharia Mecânica, Universidade Federal de Itajubá.

A detecção de danos é um importante ramo da engenharia que possibilita que medidas corretivas sejam aplicadas para garantir a segurança estrutural. O tempo de vida de qualquer estrutura pode ser predito por meio da correta determinação do dano. O objetivo do trabalho é a detecção de danos em estruturas por meio de duas técnicas, otimização global e identificação de parâmetros. A modelagem térmica do problema de detecção de danos é efetuada por meio do método de elementos de contorno. Técnicas de otimização são utilizadas para a minimização da diferença entre os valores de potencial medidos na estrutura e os valores de potencial calculados pelo programa de localização do dano. Como esta diferença é, em geral, não-convexa, os algoritmos genéticos são utilizados para resolver o problema de otimização global. Redes neurais artificiais que identificam os parâmetros desconhecidos dos danos estruturais também são utilizadas para resolver o problema inverso de detecção de danos. Esta técnica simula o comportamento não-linear entre os valores de potenciais internos na estrutura e os parâmetros do dano. Neste trabalho, uma comparação entre as diferentes técnicas é realizada e os resultados são discutidos para o problema.

Palavras-chave

Detecção de Danos, Redes Neurais Artificiais, Otimização, Algoritmos Genéticos.

Abstract

LOPES, P. S. (2007), *Damage detection on structures through artificial neural networks and genetic algorithms techniques*, Itajubá, 106p. MSc. Dissertation - Mechanical Engineering Institute, Federal University of Itajubá.

Damage detection is an important branch of engineering that allows corrective measures are applied for the structural security. Correct determinedness of damage can predict life time of every structure. The objective of this work is the structural damage detection through two techniques, global optimization and parameter identification. The thermal modeling of the damage detection problem is done with the help of the boundary element method. Optimization techniques are turned to account for the minimization of the difference between measured and computed potential values. As this difference is, in general, nonconvex, genetic algorithms are used to solve global optimization problem. Artificial neural networks that identify the unknown parameters of the structural damage are also utilized to solver the inverse problem of damage detection. This technique simulates the nonlinear behaviour between internal potential values in the structure and the damage parameters. In this work, a comparison between the different techniques is performed and the results are argued for the problem.

Keywords

Damage Detection, Artificial Neural Networks, Optimization, Genetic Algorithms.

SUMÁRIO

SUMÁRIO ..... **i**
Lista de Figuras ..... **iv**
Lista de Tabelas ..... **vi**

Simbologia ..... **vii**
Letras Latinas ..... **vii**
Letras Gregas ..... **vii**
Siglas e Abreviaturas ..... **viii**

Capítulo 1
INTRODUÇÃO **1**
1.1 REVISÃO DA LITERATURA 2
1.2 OBJETIVOS DA PESQUISA 7
1.3 CONTEÚDO 7

Capítulo 2
MÉTODOS DE OTIMIZAÇÃO **9**
2.1 INTRODUÇÃO 9
2.2 ALGORITMO GENÉTICO 10
2.2.1 Operadores genéticos 11
2.2.2 Definições básicas 13
2.2.3 Um algoritmo genético simples 13
2.2.4 Parâmetros do algoritmo genético 14
2.2.5 Algoritmo genético via Matlab® 15
2.2.6 Tratamento das restrições 16

Capítulo 3

IDENTIFICAÇÃO DE PARÂMETROS 18

3.1 INTRODUÇÃO 18

3.2 REDES NEURAIS ARTIFICIAIS 18

3.2.1 Modelo de neurônios artificiais 19

3.2.2 Arquitetura das redes neurais artificiais 20

3.2.3 Treinamento de uma rede neural artificial 21

3.2.3.1 O algoritmo Backpropagation 23

3.2.4 Redes neurais artificiais via Matlab® 25

Capítulo 4

DETECÇÃO DE DANOS 27

4.1 INTRODUÇÃO 27

4.2 MÉTODO DE ELEMENTOS DE CONTORNO PARA O PROBLEMA DE POTENCIAL 28

4.3 O PROBLEMA DE DETECÇÃO DE DANOS 30

4.3.1 Configuração do cromossomo do algoritmo genético 32

4.3.2 Configuração dos dados de entrada da rede neural artificial 32

Capítulo 5

RESULTADOS E DISCUSSÕES 34

5.1 ANÁLISE DOS RESULTADOS OBTIDOS PELO ALGORITMO GENÉTICO 34

5.2 ANÁLISE DOS RESULTADOS OBTIDOS PELA REDE NEURAL ARTIFICIAL .. 44

Capítulo 6

CONCLUSÕES E PERSPECTIVAS FUTURAS 56

6.1 CONCLUSÕES 56

6.2 PERSPECTIVAS FUTURAS 57

REFERÊNCIAS BIBLIOGRÁFICAS 59

Anexo A

PROGRAMA DO MÉTODO DIRETO POR MEIO DO MÉTODO DE ELEMENTOS DE CONTORNO 62

Anexo B

PROGRAMA DE DETECÇÃO DE DANOS UTILIZANDO ALGORITMOS GENÉTICOS 77

Anexo C

PROGRAMA DE DETECÇÃO DE DANOS UTILIZANDO REDES NEURAIS ARTIFICIAIS 85

Lista de Figuras

Figura 1 – Operadores genéticos. 11
Figura 2 – Fluxograma do GA. 14
Figura 3 – Representação do neurônio biológico 19
Figura 4 – Estrutura de um neurônio artificial (adaptado de Chong & Zak, 2001) 19
Figura 5 – Simbologia para o neurônio artificial (adaptado de Chong & Zak, 2001) 20
Figura 6 – Funções de ativação: (a) *threshold*; (b) sigmoidal; (c) linear. 20
Figura 7 – Rede neural do tipo *feedforward* 21
Figura 8 – Processo de treinamento de uma ANN (adaptado da *Neural Netwok Toolbox*) 22
Figura 9 – Rede neural BPN (adaptado de Freeman & Skapura, 1991) 23
Figura 10 – Fluxograma do BEM para o potencial 29
Figura 11 – Numeração dos elementos no BEM: (a) domínio fechado; (b) domínio aberto. . 29
Figura 12 – Distribuição dos sensores para o GA 30
Figura 13 – Distribuição dos sensores para a ANN: (a) 25, (b) 15, (c) 9, (d) 5 31
Figura 14 – Possíveis localizações dos furos na placa com 25 sensores 33
Figura 15 – Condições de contorno de uma placa quadrada (Brebbia & Dominguez, 1992) . 35
Figura 16 – Diferença de potencial normalizado para uma placa com furo central 35
Figura 17 – Discretização da placa e região da presença do furo 37
Figura 18 – Região da presença furo em 5 rodadas 38
Figura 19 – Região da presença furo para a nova configuração 39
Figura 20 – Discretização da placa e região da presença do furo na nova posição 40
Figura 21 – Região da presença furo para a nova posição 40
Figura 22 – Região de ocorrência dos dois furos encontrada pelo programa 43
Figura 23 – Resultados obtidos pela ANN para 25 sensores 45
Figura 24 – Resultados obtidos pela ANN para 15 sensores 46
Figura 25 – Resultados obtidos pela ANN para 9 sensores 47

Figura 26 – Resultados obtidos pela ANN para 5 sensores 48
Figura 27 – Novos resultados obtidos pela ANN para 25 sensores 49
Figura 28 – Novos resultados obtidos pela ANN para 15 sensores 50
Figura 29 – Novos resultados obtidos pela ANN para 9 sensores 51
Figura 30 – Novos resultados obtidos pela ANN para 5 sensores 52
Figura 31 – Resultados do programa que detecta até dois furos na placa 54

Lista de Tabelas

Tabela 1 – Resultados da ANN com 25 sensores 45
Tabela 2 – Resultados da ANN com 15 sensores 46
Tabela 3 – Resultados da ANN com 9 sensores 47
Tabela 4 – Resultados da ANN com 5 sensores 48
Tabela 5 – Resultados da ANN com 25 sensores e novos dados de entrada 50
Tabela 6 – Resultados da ANN com 15 sensores e novos dados de entrada 51
Tabela 7 – Resultados da ANN com 9 sensores e novos dados de entrada 52
Tabela 8 – Resultados da ANN com 5 sensores e novos dados de entrada 53
Tabela 9 – Precisão e tempo de execução para o primeiro caso 53
Tabela 10 – Precisão e tempo de execução para o segundo caso 53
Tabela 11 – Resultados do programa que detecta até dois furos na placa 55

Simbologia

Letras Latinas

F	função objetivo do problema penalizado
P	função de penalização
f, g, h	Representação de funções
x	abscissa de um ponto *[m]*
y	ordenada de um ponto *[m]*
n	número de restrições de desigualdade
l	número de restrições de igualdade
cm	centímetros
m	metros
s	segundo
xc	coordenada x do centro do furo *[m]*
yc	coordenada y do centro do furo *[m]*
r	raio do furo *[m]*
J	valor do funcional

Letras Gregas

ρ	fator de penalização

Siglas e Abreviaturas

ANN’s	Redes Neurais Artificiais
BEM	Método de Elementos de Contorno
BPN	Rede Neural *Backpropagation*
EKF	Filtro de Kalman Estendido
GA	Algoritmo Genético
IEM	Instituto de Engenharia Mecânica
KF	Filtro de Kalman
LC-BEM	BEM e a estratégia de problema de complementaridade linear
SQP	Programação Seqüencial Quadrática

Capítulo 1

INTRODUÇÃO

A detecção de danos é um importante ramo da engenharia que possibilita que medidas corretivas sejam aplicadas para garantir a segurança estrutural. O tempo de vida de qualquer estrutura pode ser predito por meio da correta determinação do dano. Esta correta determinação do dano possibilita avaliar a integridade da estrutura, facilitando uma possível intervenção na mesma, caso seja necessário.

O problema de detecção de danos pode ser classificado como um problema de identificação de sistemas ou um problema inverso. Neste tipo de problema, técnicas de identificação de parâmetros, como redes neurais artificiais (ANN) e filtro de Kalman (KF), podem ser utilizadas para determinar os parâmetros desconhecidos do dano. Técnicas de otimização procuram minimizar ou maximizar uma determinada função com o propósito de encontrar a melhor solução dentro de um conjunto de possíveis soluções. Os algoritmos genéticos (GA's) pertencem à categoria de otimização global onde o ótimo global do sistema possui maiores chances de ser obtido. Neste método, o dano é encontrado pela minimização de um funcional. Este funcional pode ser definido como a diferença entre os valores medidos ou simulados da diferença de potencial (entre a placa sem dano e a placa com dano) e os calculados pelo programa de detecção de danos.

Neste trabalho, um exemplo de transferência de calor por condução simples numa placa fina é estudado e o método de elementos de contorno (BEM) é utilizado para obter os valores de potencial na placa, tanto na placa sem dano quanto na placa com dano. Por falta de dados

obtidos experimentalmente, os valores medidos da diferença de potencial entre a placa sem danos e a placa com dano foram simulados pelo BEM. Os valores de potencial sobre a superfície externa da placa representam a distribuição de temperaturas sobre a mesma. Como suposição, a condução de calor por possíveis furos presentes no interior da placa é considerada nula (os furos são considerados adiabáticos). O uso de técnicas térmicas mostra que a distribuição de temperaturas sobre uma placa muda devido às variações nas propriedades mecânicas da mesma, o que poderia estar relacionado a um determinado dano (Cecchini, 2005).

A resolução de problemas por meio de métodos analíticos muitas vezes é uma tarefa difícil, ou até mesmo impossível de ser obtida. O uso de técnicas de otimização global possibilita investigar problemas onde a função objetivo e suas respectivas restrições são não lineares ou descontínuas. Já técnicas de identificação de parâmetros, promovem uma identificação em tempo real dos parâmetros desconhecidos do problema e tentam relacionar as informações dadas como entradas com as informações fornecidas na saída. Então, a principal motivação para o desenvolvimento deste trabalho é o estudo de duas técnicas que possibilitam encontrar de maneira distinta a presença de danos ou defeitos por meio da transferência de calor por condução simples numa placa fina. Dano é definido como mudanças nas propriedades ou na geometria dos materiais, como mudanças nas condições de contorno, e que afetam o desempenho desses materiais. Alguns exemplos de danos que podem estar presentes numa determinada estrutura são os furos, representando a localização de possíveis rebites ou parafusos nas estruturas, e as trincas.

1.1 REVISÃO DA LITERATURA

No trabalho de Stravoulakis & Antes (1998), a identificação de defeitos únicos tanto quanto de múltiplos defeitos é considerada. O método de elementos de contorno é utilizado para modelar numericamente o problema mecânico direto e técnicas de otimização numérica são utilizadas para resolver o problema de identificação. O artigo cita algumas publicações a respeito de identificação de defeitos, falhas e trincas, então, descreve a formulação do BEM para a solução do problema direto, a formulação do problema inverso, a solução numérica do problema inverso pelo uso de otimização e apresenta as discussões dos resultados. O método da programação seqüencial quadrática (SQP) é utilizado para determinar o ótimo local do erro

(diferença entre o valor medido e a resposta desejada) e o GA, para determinar o ótimo global da mesma função. O artigo fornece ainda dois exemplos numéricos: uma placa sob carregamento estático com três furos circulares e a mesma placa sob carregamento harmônico dinâmico. Por meio destes exemplos verifica-se que um algoritmo de otimização global é a melhor escolha para a solução do problema de minimização.

Liang & Hwu (2001) aplicam uma ANN para a identificação on-line de furos/trincas em estruturas compósitas. A ANN utiliza duas camadas internas (*hidden*) para simular o relacionamento não linear entre as deformações medidas e os parâmetros de furos ou trincas (o tamanho, a localização e a orientação de danos na estrutura). As deformações são encontradas por meio do BEM, considerando três diferentes modos de carregamento para cada sensor. O algoritmo de treinamento *backpropagation* é utilizado e este algoritmo possui três estágios: entrada de dados (deformações medidas, tamanho, localização e orientação de furos/trincas), treinamento e teste. Por meio da análise dos resultados, a melhor configuração para a rede neural é encontrada, ou seja, os resultados são analisados para diferentes tipos de funções de transferências, diferentes arquiteturas da camada interna (*hidden*), diferentes padrões de treinamento, a adição de erros presentes nas medidas, diferentes números de sensores e com furos de diferentes formas.

Burczynski & Beluch (2001) formulam o problema de identificação de trinca como a minimização da diferença entre os valores medidos e os computados de deslocamentos ou tensões em nós selecionados do contorno. A formulação do problema de identificação de trincas é apresentada por meio de um funcional que representa a norma entre os valores teóricos e os computados de deslocamentos. O problema pode ser resolvido por meio da minimização deste funcional. O BEM é usado para encontrar a função objetivo e algoritmos evolucionários são utilizados na identificação de trincas. A estrutura do cromossomo no algoritmo evolucionário é construída de diferentes modos: a identificação de uma única trinca e múltiplas trincas considerando que o número de trincas é conhecido; a identificação de múltiplas trincas considerando que o número de trincas é desconhecido; e, a identificação de trincas com a perturbação estocástica das quantidades medidas. Dois métodos híbridos de identificação também são apresentados: o modelo linear e o modelo paralelo. A vantagem da abordagem híbrida é a redução no tempo gasto em cálculos gerados no algoritmo evolucionário.

Liu et al. (2002) propõem um método de análise inversa usando a rede neural *backpropagation* (BPN) e a mecânica computacional, combinando o método de elementos

finitos com a equação integral de contorno. No artigo, a BPN e a mecânica computacional simulam um teste ultra-sônico não destrutivo *A-scan*. No problema direto, respostas em freqüência de uma trinca média sob impacto são calculadas pela mecânica computacional. No problema inverso, a BPN pode aprender o mapeamento entre as entradas e as saídas por meio de um conjunto de dados simples e determinar a classe de novos dados baseada no conhecimento prévio. Esta rede é treinada pelos parâmetros extraídos de várias superfícies de respostas obtidas do problema direto. Então, após o treino, a rede é utilizada para a classificação e identificação de trincas médias para determinar o tipo, a localização e o comprimento da trinca. Em geral, o sucesso de qualquer ANN depende da escolha das entradas. Para construir a base de dados com dados experimentais para o treino da rede, muitos experimentos devem se realizados. Portanto, simulações numéricas por meio da mecânica computacional são mais flexíveis e mais rápidas do que a experimentação para fins práticos.

Douka et al. (2003) estimam a localização e o tamanho de uma trinca em uma viga engastada. A transformada contínua de *Wavelet* é utilizada para analisar o modo de vibração fundamental desta viga. O artigo cita também as vantagens e desvantagens de outros dois métodos de identificação de danos, freqüências naturais e modos naturais. A transformada *Wavelet* foi utilizada no trabalho pelo motivo de não requerer diferenciação numérica dos dados medidos e apresentar uma melhor precisão na detecção de anomalias. Uma investigação analítica e experimental de uma viga engastada com uma trinca de superfície transversal é realizada para validar o método proposto. Os efeitos causados por ruídos nos resultados também são analisados. Os resultados apresentados podem ser estendidos a estruturas e condições de contorno mais complexas.

Segundo Simon (2001), o KF é uma ferramenta que pode estimar as variações de uma ampla classe de processo e é o único que minimiza a variação do erro de estimação. O artigo fornece os conceitos básicos necessários para a implementação do KF e fornece, ainda, um exemplo de aplicação. Uma modelagem matemática de um sistema linear é feita para então apresentar as equações do KF. O código escrito em Matlab® para o exemplo dado é apresentado e dicas de como obter códigos escritos na linguagem C para manipular matrizes e o KF são fornecidas. O artigo discute a respeito do filtro de Kalman estendido (EKF) para sistemas não lineares e apresenta o filtro "H∞" que constitui uma alternativa ao KF. Este filtro "H∞" é eficiente em situações onde o ruído de medida e o de processo é dependente um do outro e é útil na minimização do "pior" erro de estimação. O filtro "H∞" é eficiente ainda

quando as matrizes de covariância de ruído não são conhecidas. Neste trabalho, o autor fornece uma perspectiva histórica, explicando que algumas vezes o KF é conhecido como filtro de Kalman-Bucy. Enfim, o KF surgiu para resolver um problema específico e desde então tem encontrado aplicações em diversas áreas.

Uma introdução a respeito do KF é feita por Welch & Bishop (2004), sendo que o filtro é constituído por um conjunto de equações matemáticas que fornecem um eficiente método computacional pra estimar os estados de um processo, em um modo que minimiza a média do erro quadrado. No trabalho, primeiramente é realizada a modelagem matemática do KF discreto, para em seguida realizar a modelagem matemática do EKF. O EKF é uma derivação do filtro tradicional e é utilizado quando o processo for não linear. O algoritmo possui uma seqüência a ser seguida, primeiro devem ser calculadas as equações de tempo, encontrando as estimativas "a priori", depois, calcular as equações de média, encontrando as estimativas "a posteriori". O processo é repetido com a estimativa "a posteriori" sendo a nova estimativa "a priori" nas equações de tempo.

Segundo Lages (2004b), o KF é um estimador linear recursivo que calcula uma estimativa de variância mínima para um estado que evolui no tempo a partir de observações relacionadas linearmente com este estado. No artigo é feita a modelagem matemática do KF, obtendo expressões que podem ser representadas de diversas formas. A forma de covariância inversa e a forma de informação do KF são discutidas. Em seguida, o artigo trata de problemas que podem surgir no KF, tais como, erros de arredondamento, erros de modelagem que levam o filtro a ajustar uma curva errada aos dados experimentais, problemas de observabilidade onde uma ou mais variáveis de estado podem não ser observáveis e, ainda, situações práticas que podem levar o algoritmo a divergir. Desta forma, o EKF é apresentado, sendo uma modificação do filtro tradicional para utilização em sistemas não lineares. Finalmente, o filtro de informação estendido que sofre os mesmos problemas de instabilidade que o EKF são discutidos.

Troncoso (2004) discute a estimação de parâmetros para sua aplicação à detecção e diagnóstico de falhas. O trabalho começa apresentando os fundamentos de identificação de sistemas e estimação de parâmetros. Em seguida, é apresentada a estimação em sistemas dinâmicos lineares invariáveis e variáveis. Então, a estimação em sistemas dinâmicos não lineares é apresentada. Com esta introdução teórica, métodos de estimação em sistemas dinâmicos lineares variáveis são comparados, passando para os fundamentos de detecção e diagnóstico de falhas. O autor afirma que os modelos lineares, KF e KF competitivo,

apresentam um melhor comportamento dentro e fora de linha, respectivamente. Por fim, é discutido sobre o KF e sobre algumas de suas variações como o KF em estado estacionário e o EKF.

No trabalho de Faria & Souza (2004) o KF foi utilizado num problema de rastreamento. Os autores observaram que, para os resultados serem confiáveis, uma ponderação era necessária ser feita entre as medidas obtidas e as previsões feitas pela teoria por meio do KF. Os autores utilizaram os parâmetros de um veículo (posição, velocidade e o ângulo que este fazia com um segundo veículo) para estimar a trajetória do segundo veículo com velocidade muito menor em relação ao primeiro. Caso o ângulo não fosse obtido, uma predição era feita para estimar as posições futuras. Deste segundo veículo, que se desejava descobrir a trajetória, não se tinha conhecimento nem da massa nem da dinâmica, então, apenas equações cinemáticas eram consideradas. A qualidade do procedimento em tempo real foi averiguada e, por meio de uma dada tolerância, a diferença entre o valor simulado para um caso real e um valor estimado foi observada. Caso fossem necessários, parâmetros do processo eram escolhidos a fim de melhorar os resultados obtidos. Por fim, o KF foi implementado em Matlab®, possibilitando a simulação da posição e da velocidade horizontais do segundo veículo.

O KF é utilizado por Silva & Pereira (2005) para realizar uma estimativa de posição continuadas de um robô móvel utilizando quatro sensores. No trabalho os autores discutem a respeito das características do robô que se move em ambientes estáticos. Os resultados dos testes executados são apresentados por uma técnica de localização por KF. E para facilitar a integração com o simulador, a linguagem Java é utilizada. No trabalho é informada como a implementação para a simulação foi efetuada e como se comporta a arquitetura do simulador. Com isto, dos resultados obtidos foram analisados que em trajetórias sem mudança brusca de direção, a estimativa se aproxima mais da real à medida que obstáculos eram encontrados. Finalmente, é concluído que o KF é útil para reduzir os erros acumulados por sensores de odometria, mas em trajetórias longas e sinuosas, é necessária a leitura dos sensores como auxílio.

Engelhardt et al. (2006) utilizam o EKF na solução numérica de problemas de identificação de trincas e falhas. A modelagem mecânica do problema elasto-dinâmico é auxiliada pelo BEM. As trincas são modeladas por elementos hipersingulares e o contato unilateral (abertura e fechamento de trinca; sem atrito) por uma combinação de BEM e a estratégia de problema de complementaridade linear (LC-BEM). As equações do KF foram

manipuladas algebricamente, evitando desta forma dificuldades numéricas que surgem no KF regular. Uma trinca retilínea pode ser parametrizada por meio das coordenadas de seu centro, o seu comprimento e a sua orientação. Um furo elíptico pode ser parametrizado pelo seu centro, sua orientação e os dois semi-eixos (maior e menor). Estes parâmetros são as variáveis que devem ser estimadas. Após toda a discussão a respeito da formulação usada no artigo para a identificação de defeitos, os resultados numéricos para um disco são apresentados. As influências da forma do defeito, do tipo de carregamento, do contato unilateral e da forma do disco nos resultados são analisadas. O artigo fornece também como se deve proceder para realizar a identificação de quatro trincas no interior do disco.

1.2 OBJETIVOS DA PESQUISA

Os principais objetivos da pesquisa são:

- a detecção de danos estruturais, tais como, a detecção de furos circulares em placas finas que afetam o comportamento das mesmas;
- a otimização do sistema de localização do dano por meio da comparação entre os valores medidos (simulados) e os computados pelo programa de detecção de danos, usando algoritmos genéticos (GA's);
- a identificação de parâmetros dos danos (a localização e o raio do furo), por meio das redes neurais artificiais (ANN's);
- comparação entre os resultados encontrados pelas duas técnicas apresentadas (otimização e identificação de parâmetros), mostrando qual a mais adequada ao problema de detecção de danos nos casos estudados.

1.3 CONTEÚDO

Este trabalho está dividido em seis capítulos e 3 anexos. No Capítulo 1, algumas considerações iniciais sobre o trabalho são apresentadas e em seguida, são listados os

principais objetivos da pesquisa. No Capítulo 2 é descrito o GA que consiste de um método de otimização global baseado em heurísticas. Este método é utilizado para a determinação de danos presentes nas estruturas. No Capítulo 3 é descrito um método de identificação de parâmetros conhecido como ANN's. No Capítulo 4 é contextualizado o problema de detecção de danos e como proceder para resolvê-lo. Já no Capítulo 5 são apresentados os resultados obtidos para a localização do dano e, também, são feitas as análises dos mesmos. Finalmente, no Capítulo 6, algumas conclusões e sugestões para trabalhos futuros são feitas. No Anexo A está a listagem completa do programa desenvolvido em Matlab® para o BEM para o problema do potencial. No Anexo B e C estão os programas de detecção de danos por meio do GA e por meio das ANN's, respectivamente.

Capítulo 2

MÉTODOS DE OTIMIZAÇÃO

2.1 INTRODUÇÃO

A otimização pode ser definida como um processo de busca da melhor solução dentro de um conjunto de possíveis soluções. Existem dois grandes grupos em que os métodos de otimização se dividem:

- os métodos de otimização local baseados em cálculo, como a programação seqüencial quadrática (SQP); e
- os métodos de otimização global baseados em heurísticas, como o algoritmo genético (GA).

O primeiro grupo de métodos geralmente é empregado em problemas que apresentam apenas um extremo no intervalo considerado, também conhecidos como problemas unimodais. Em problemas multimodais, estes métodos dependerão da escolha do ponto inicial, podendo convergir para o extremo local mais próximo da direção de busca determinada pelas derivadas. Assim, em problemas multimodais é mais freqüente o uso de métodos do segundo grupo, pois por meio destes métodos o extremo (ótimo) global do sistema possui maiores chances de ser obtido.

Nos problemas de otimização, restrições sobre as variáveis do problema podem ou não existir. O problema é definido como um problema de otimização restrita quando a função objetivo a ser minimizada ou maximizada está sujeita a restrições. De outro modo, o problema é definido como um problema de otimização irrestrita.

Ao contrário dos métodos clássicos, os métodos de otimização global baseados em heurísticas não desenvolvem a busca a partir de um único ponto, mas a partir da geração de uma população de pontos iniciais que representam as possíveis soluções do problema.

2.2 ALGORITMO GENÉTICO

O GA é um método de buscas baseadas nos processos de evolução natural. Este método trabalha com um conjunto de possíveis soluções para um dado problema, constituindo a população inicial. Neste algoritmo as variáveis do problema são representadas como genes em um cromossomo, também denominado indivíduo. Partindo de uma população inicial, os indivíduos com características genéticas melhores adaptadas possuem maiores chances de sobreviverem e de se reproduzirem.

Segundo Burczynski & Beluch (2001), os GA's são métodos que não dependem da escolha do ponto inicial, aumentando as chances de se obter o ótimo global do sistema. Para que a população se diversifique e mantenha determinadas características de adaptação adquiridas pelas gerações anteriores, os operadores genéticos seleção, cruzamento e mutação são utilizados. Estes operadores transformam a população através de sucessivas gerações, estendendo a busca até chegar a um resultado satisfatório. A Figura 1 mostra de modo geral como estes operadores genéticos podem ser empregados.

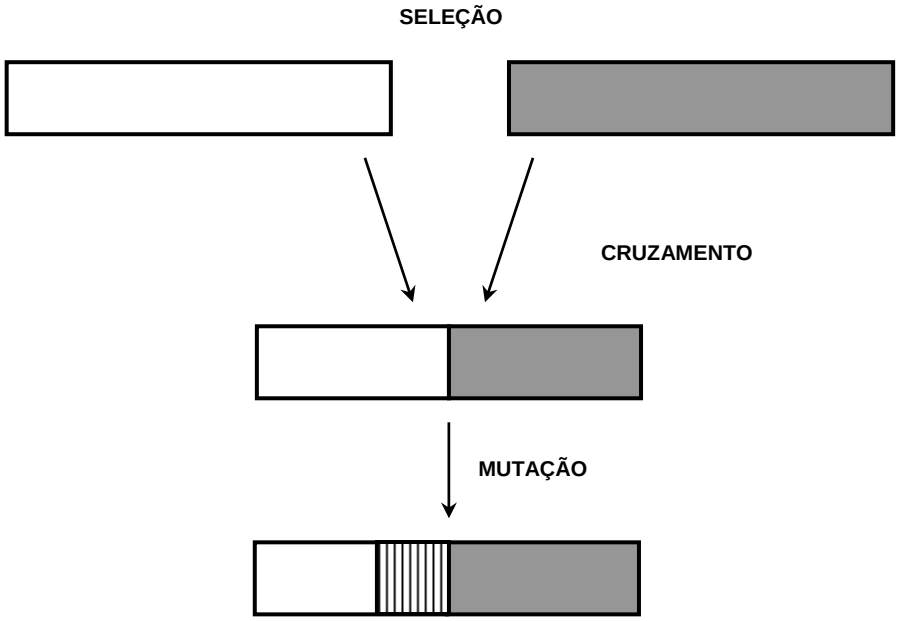

Figura 1 – Operadores genéticos.

2.2.1 Operadores genéticos

- ***Seleção:***

O operador seleção é uma versão artificial do processo de seleção natural (Goldberg, 1989). Neste tipo de operador há a seleção dos indivíduos mais aptos da geração atual que são escolhidos para a próxima geração. A função que calcula a aptidão de cada indivíduo na população é conhecida como função de avaliação ou função *fitness*. Cada indivíduo da população atual de cromossomos é avaliado quanto sua aptidão e um subconjunto de cromossomos é selecionado para ser usado como pais das próximas gerações.

Existem diversas técnicas de seleção, dentre elas há a seleção do tipo roleta (amostragem estocástica com substituição) que é a técnica mais difundida e utilizada (Spall, 2003). Neste tipo de seleção uma das seções da roleta (correspondendo a um indivíduo) é selecionada com uma probabilidade igual à área da seção. Outra técnica de seleção utilizada é a estocástica uniforme (usada como padrão pela *Genetic Algorithm and Direct Search Toolbox* do Matlab®), onde os indivíduos da geração atual são escolhidos de forma aleatória para a reprodução. Existem ainda as técnicas de torneio e a uniforme que podem ser utilizadas na seleção dos indivíduos.

Associado ao processo de seleção há a estratégia de elitismo que auxilia na melhora da convergência do GA. O elitismo consiste em manter um dado número de indivíduos em cada geração e estes indivíduos são passados diretamente à próxima geração, garantindo a preservação destes indivíduos (Mitchell, 1999; Spall, 2003).

Para Mitchell (1999), o processo de seleção deve ser combinado com os processos de cruzamento e de mutação. Estes dois operadores, cruzamento e mutação, estão descritos a seguir.

- ***Cruzamento:***

Considerado o operador genético dominante, é utilizado para gerar uma nova população por meio da recombinação de soluções (cromossomos). Um par de indivíduos é dividido em locais escolhidos aleatoriamente e seus materiais genéticos são recombinados formando novos indivíduos, que novamente são avaliados e recebem um novo valor de aptidão individual.

Dentre as técnicas de cruzamento tem-se, o cruzamento de um ponto, o de dois pontos ou múltiplos pontos, o espalhado (*scattered*) e o de heurísticas. Na primeira técnica um ponto de cruzamento é escolhido aleatoriamente (maior que zero e menor que o número de genes) e a partir deste ponto as informações genéticas dos pais são trocadas, formando desta forma dois filhos. O cruzamento de dois pontos ou múltiplos pontos corresponde a uma generalização da técnica de cruzamento de um ponto. Nesta técnica, dois ou mais pontos são escolhidos aleatoriamente e as informações contidas entre esses pontos de corte são trocadas pelo casal, formando assim dois novos filhos. No cruzamento espalhado (*scattered*) (padrão da *Genetic Algorithm and Direct Search Toolbox* do Matlab®), um vetor aleatório binário é selecionado, atribuindo ao primeiro pai o valor 1 e ao segundo o valor 0. No cruzamento por heurística, um único filho é produzido de dois pais (Burczynski & Beluch, 2001).

A escolha de qual técnica de cruzamento é utilizada depende do problema que está sendo analisado. Uma determinada técnica pode ser eficiente a um problema e ineficiente a outro.

- ***Mutação:***

Este operador é utilizado para fornecer novas informações para as populações, ou seja, o operador mutação promove uma diversidade genética uma vez que a população inicial pode

ser insuficiente de informações para encontrar a solução (Spall, 2003). Com o uso deste operador, uma maior varredura do espaço de busca é feita, evitando-se assim que o algoritmo genético convirja para mínimos locais.

Existem as técnicas de mutação uniforme onde cada gene de um cromossomo possui exatamente a mesma chance de sofrer mutação, e a mutação gaussiana (normal) onde um número aleatório de uma distribuição normal com média zero é adicionado à entrada do vetor de indivíduos usados para gerar as próximas gerações.

2.2.2 Definições básicas

- **Função de avaliação (*fitness*):** função que avalia a aptidão de cada indivíduo em cada geração do processo;
- **Gene:** representação de cada parâmetro (variável) da solução. Registram as características dos indivíduos e são responsáveis por transmiti-las a seus descendentes;
- **Genótipo:** constituição genética do indivíduo. Nos algoritmos genéticos, ele é responsável pela distribuição dos genes num cromossomo;
- **Fenótipo:** cromossomo codificado;
- **Cromossomo:** formado por um conjunto de genes, representando uma possível solução para o problema.
- **População:** conjunto de cromossomos ou soluções no espaço de busca;
- **Geração:** iteração completa do GA que gera uma nova população;
- **Operação genética:** operações que o GA realiza sobre cada um dos cromossomos;
- **Espaço de busca (região viável):** compreende as possíveis soluções do problema a ser otimizado e é caracterizado pelas restrições impostas ao problema.

2.2.3 Um algoritmo genético simples

Existem diversas abordagens a respeito de como um GA pode ser implementado. Mitchell (1999), em seu livro, apresenta como um GA simples trabalha:

1. Gerar aleatoriamente uma população de cromossomos (possíveis soluções para o problema);
2. Calcular a aptidão de cada indivíduo na população, utilizando a função de avaliação;

3. Selecionar os cromossomos da população atual que formarão os descendentes da próxima geração. Aplicar os operadores cruzamento e mutação sobre os cromossomos selecionados para criar a próxima geração de cromossomos;
4. Substituir a população atual pela nova população gerada;
5. Se o método convergir, termine o processo e retorne o melhor indivíduo gerado. Caso não convirja, volte ao *passo 2*.

A cada iteração, uma nova população é criada e esta nova população deve representar uma melhor aproximação da solução do problema de otimização. O algoritmo converge quando algum critério de parada é verificado. O fluxograma na Figura 2 representa o GA simples.

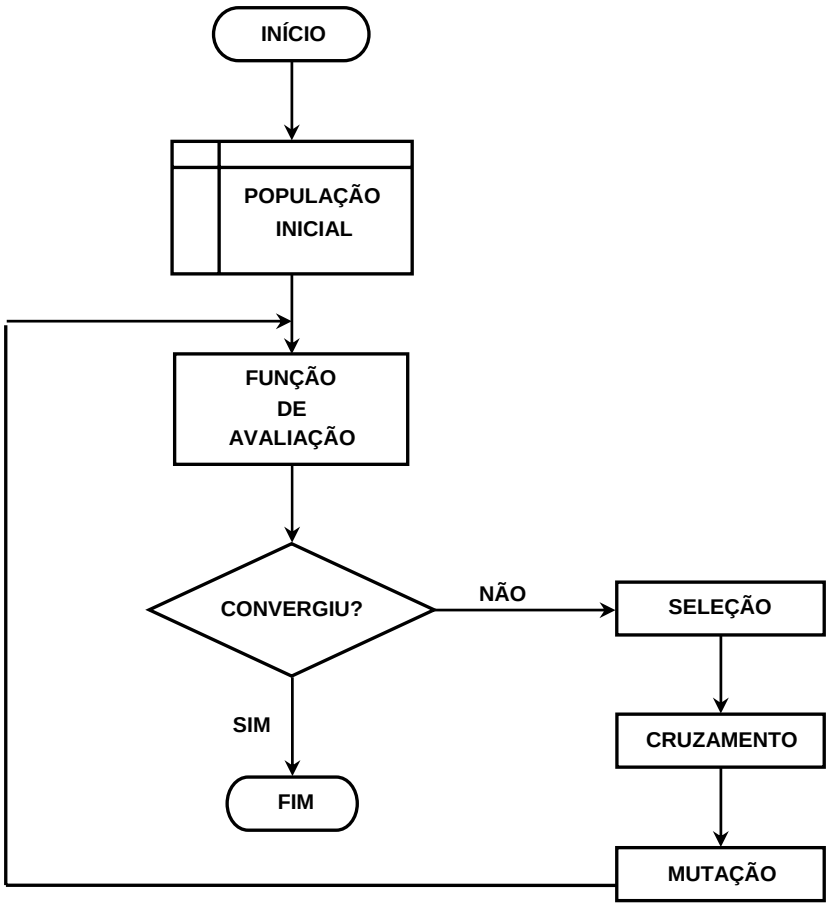

Figura 2 – Fluxograma do GA.

2.2.4 Parâmetros do algoritmo genético

Um cromossomo pode ser representado como uma cadeia de bits, um vetor com números de dupla precisão (*double*) ou outro tipo de representação que dependerá do

problema em estudo. Existem vários parâmetros do GA que influenciam no comportamento do método. Os parâmetros mais importantes são: o tamanho da população, o número de gerações, a probabilidade de cruzamento (*crossover*) e a probabilidade de mutação. A influência de cada parâmetro no desempenho do algoritmo também dependerá do problema de aplicação. A tarefa de escolha da melhor configuração para os parâmetros é árdua e dependerá da realização de um grande número de experimentos e testes.

O desempenho global do GA é influenciado pelo tamanho da população. Este parâmetro indica o número de cromossomos que há em cada população, definindo o espaço de busca do problema. Quando este parâmetro é alto, mais o espaço de busca do problema é varrido, mas apresentará um alto custo computacional para realizar muitas avaliações da função de aptidão. Por outro lado, quando o parâmetro é pequeno, o desempenho do algoritmo diminui.

2.2.5 Algoritmo genético via Matlab®

O software Matlab® possui a *Genetic Algorithm and Direct Search Toolbox* que tem a função *ga* responsável por encontrar o mínimo de uma função usando o GA. A configuração dos parâmetros do GA pode ser fornecida por meio da função *gaoptimset*. Então, diferentes opções de configuração podem ser feitas de forma a adequá-las a cada problema.

Alguns critérios de parada disponíveis para o GA são:

- máximo número de gerações;
- tempo máximo para o GA rodar antes de parar;
- parada do GA quando não há melhoramento do melhor valor de aptidão para o número de gerações; ou
- parada do GA quando não há melhoramento do melhor valor de aptidão para um dado intervalo de tempo.

Existem outros parâmetros que podem ser configurados, tais como, a função de cruzamento (*CrossoverFcn*), a função de mutação (*MutationFcn*), a taxa de cruzamento (que se relaciona com a taxa de mutação, sendo esta última de valor menor), a população inicial, o número de gerações, o tamanho da população, entre outras. A função *gaoptimset* fornece também os tipos de populações (*PopulationType*), ou seja, o tipo de dados de entrada que o

GA suporta, tais como, *double*, *bit* ou algum outro tipo de população definido pelo usuário. Enfim, a configuração dos parâmetros do GA depende do problema sob estudo.

2.2.6 Tratamento das restrições

Embora o GA seja um método irrestrito, problemas restritos podem ser transformados em problemas irrestritos por meio de métodos de penalização. Existem dois métodos de penalização, o método das barreiras e o método de penalidades. O método de barreiras impõe uma penalidade para o alcance do contorno de uma restrição de desigualdade, enquanto o método de penalidades impõe uma penalidade para a violação de uma restrição. Ao contrário do método das barreiras, no método das penalidades o ponto de partida do método não tem necessidade de estar na região viável do problema (Nash & Sofer, 1996).

O método das penalidades transforma problemas restritos em outros irrestritos pela adição de uma função de penalidade com as restrições violadas, sendo estas restrições de igualdade ou desigualdade.

O problema geral de otimização (restrita) é dado conforme a equação (1)

$$\begin{aligned} &\min_{x \in R^n} f(x) \\ &s.a\ g_j(x) \le 0 \quad j = 1,...,n \\ &\quad h_i(x) = 0 \quad i = 1,...,l \end{aligned} \tag{1}$$

Transformando o problema apresentado na equação (1) em um problema irrestrito, a nova função objetivo do problema penalizado possui a forma da equação (2)

$$F(x,\rho) = f(x) + 0{,}5\,\rho\,P(x) \tag{2}$$

onde ρ corresponde ao fator de penalização imposto à violação das restrições do problema e $P(x)$ a função de penalização dada conforme a equação (3):

$$P(x) = \sum_{j=1}^{n} \left\{ min\left[0, -g_j(x)\right] \right\}^2 + \sum_{i=1}^{l} \left[h_i(x) \right]^2 . \tag{3}$$

O primeiro somatório indica que deve ser levado em conta o menor valor entre 0 e $-g_j(x)$. Se há a violação da restrição imposta ao problema, ou seja, $g_j(x) > 0$, a restrição violada é penalizada por um fator de $0{,}5\rho$. O segundo somatório corresponde à violação das restrições de igualdade.

Capítulo 3

IDENTIFICAÇÃO DE PARÂMETROS

3.1 INTRODUÇÃO

A identificação de parâmetros pode ser feita por meio de duas técnicas, as redes neurais artificiais (ANN's) e o filtro de Kalman (KF). Ambas as técnicas simulam o comportamento não linear entre os valores de potencial medidos na placa e os parâmetros do furo, ou seja, a localização e o raio do furo. Neste trabalho, apenas as ANN's foram discutidas.

Segundo Liang and Hwu (2000), a otimização não-linear não é capaz de realizar uma identificação em tempo-real, mas uma ANN consegue executar esta tarefa. No tópico a seguir é discutido a respeito da ANN.

3.2 REDES NEURAIS ARTIFICIAIS

As ANN's são formadas por unidades menores chamadas neurônios que estão conectados uns aos outros através de sinapses. Um neurônio típico, como pode ser visualizado na Figura 3, é composto por um corpo celular ou soma, um axônio e várias ramificações conhecidas como dendritos. Os dendritos são os terminais de entrada do neurônio e o axônio é um longo terminal de saída responsável pela transmissão de informações. O núcleo que

guarda toda a informação genética e está presente no corpo celular (soma). A comunicação entre neurônios é feita por meio de sinapses, ou seja, a comunicação é feita na região de contato entre dois neurônios por meio da transmissão de impulsos nervosos entre eles. Neste contexto, as ANN's são técnicas computacionais que apresentam um modelo matemático para representar o cérebro humano e tentar simular o seu processo de aprendizagem.

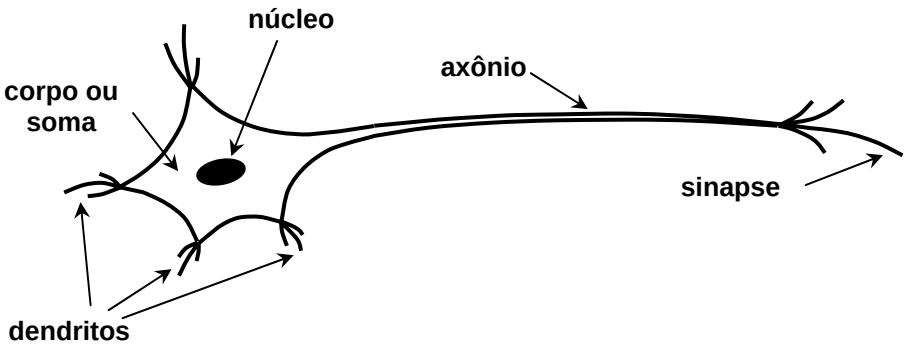

Figura 3 – Representação do neurônio biológico

3.2.1 Modelo de neurônios artificiais

O neurônio artificial representa um modelo simplificado do neurônio biológico. Conforme pode ser visualizado na Figura 4, x_1 a x_n representam os *n* terminais de entrada (dendritos), y_1 a y_m, os *m* terminais de saída, w_{1j} a w_{nj} são as ponderações nas entradas representando as sinapses entre os neurônios, e a função de ativação (*threshold function*) representa a função na saída do neurônio (ativação ou inibição do neurônio). Cada sinal de entrada é multiplicado por um peso indicando a influência destes sinais na saída do neurônio. Então, uma soma ponderada é feita, produzindo um nível de atividade, se este nível exceder um dado limite (*threshold*), a informação é passada para outros neurônios. Neste caso, o neurônio está ativo. A Figura 5 apresenta a simbologia para o neurônio da Figura 4.

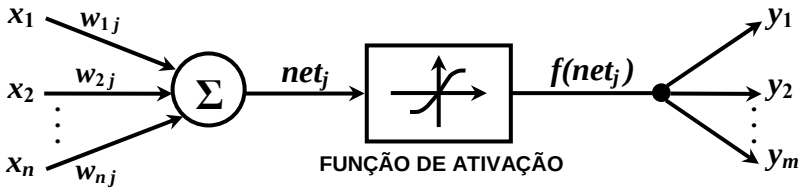

Figura 4 – Estrutura de um neurônio artificial (adaptado de Chong & Zak, 2001)

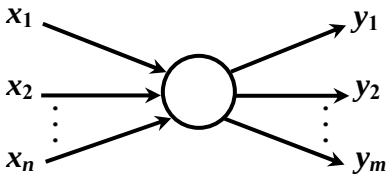

Figura 5 – Simbologia para o neurônio artificial (adaptado de Chong & Zak, 2001)

Conforme pode ser visualizado pela Figura 4, o somatório de todas as entradas do neurônio *j* multiplicadas pelos seus respectivos pesos é dado pela equação (4):

$$net_j = \sum_{i=1}^{n} x_i \, w_{ij} \tag{4}$$

sendo que w_{ij} representa o peso da sinapse do neurônio *i* para o neurônio *j*. O valor resultante do somatório representa o *net* do neurônio *j* e este valor é aplicado na entrada da função de ativação.

A função de ativação $f(.)$ pode ser do tipo *threshold* (Figura 6(a)), sigmoidal (Figura 6(b)), linear (Figura 6(c)), ou outra. Para uma função de ativação *threshold*, a saída é configurada para um de dois níveis, dependendo se a entrada é maior ou menor do que um valor limite (*threshold*). Para uma função de ativação sigmoidal, a saída varia continuamente, mas não linearmente como as mudanças nas entradas. Finalmente, para uma função de ativação linear, a saída é proporcional à saída ponderada total.

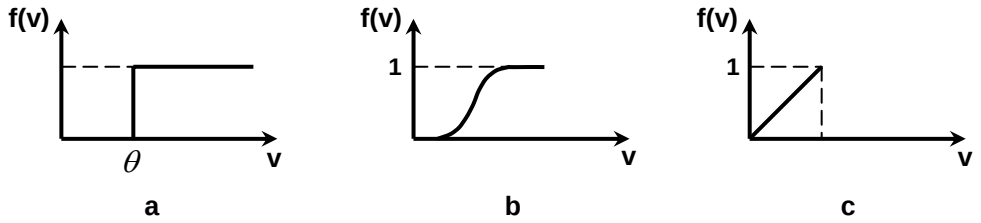

Figura 6 – Funções de ativação: (a) *threshold*; (b) sigmoidal; (c) linear.

3.2.2 Arquitetura das redes neurais artificiais

Uma ANN é formada pelos neurônios interconectados cujas entradas ou são obtidas das saídas de outros neurônios ou de nós de entrada. Diferentes configurações do neurônio artificial podem ser feitas para desenvolver diferentes configurações ou topologias das redes

(Rao et al., 2005). As topologias de redes podem ser definidas pelo número de camadas, quantidade de neurônios nas camadas e pelo tipo de conexão entre os neurônios. Dentre as configurações existentes, a ANN pode ser do tipo *feedforward* (direta) ou do tipo *feedback* (recorrente). Nas redes neurais do tipo *feedforward*, os neurônios são interconectados em camadas, mas o fluxo de dados ocorre em apenas uma direção, ou seja, não há a realimentação (Chong & Zak, 2001). Nas redes neurais do tipo *feedback*, há pelo menos um ciclo de realimentação, ou seja, um neurônio recebe a informação tanto de neurônios da camada anterior quanto de uma camada posterior.

A primeira camada na rede é denominada camada de entrada (*input layers*), a última camada é denominada camada de saída (*output layers*) e as camadas existentes entre estas duas camadas, são as camadas intermediárias ou ocultas (*hidden layers*). Problemas mais complexos podem ser implementados devido ao uso das camadas intermediárias, porém o aprendizado da ANN se torna mais difícil. A Figura 7 ilustra uma rede neural do tipo *feedforward* com três neurônios na camada de entrada, duas camadas intermediárias, e dois neurônios na camada de saída.

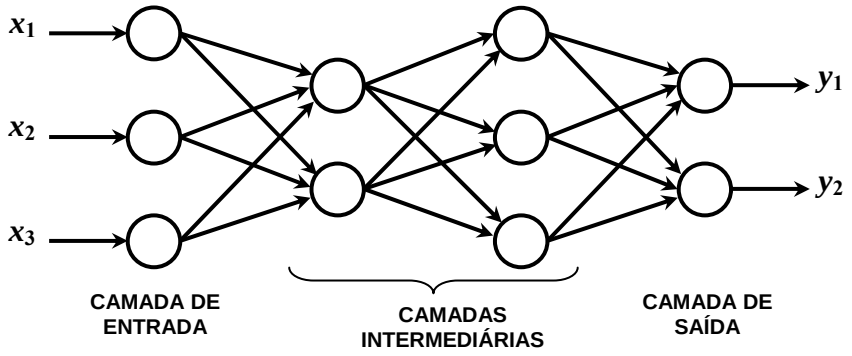

Figura 7 – Rede neural do tipo *feedforward*

3.2.3 Treinamento de uma rede neural artificial

O treinamento é um processo iterativo de ajuste dos pesos de uma ANN. Uma ANN aprende quando uma solução generalizada para uma classe de problemas é alcançada, ou seja, até que uma dada entrada conduza a um valor de saída especificado (*target output*). A Figura 8 mostra como é feito o processo de treinamento e aprendizagem.

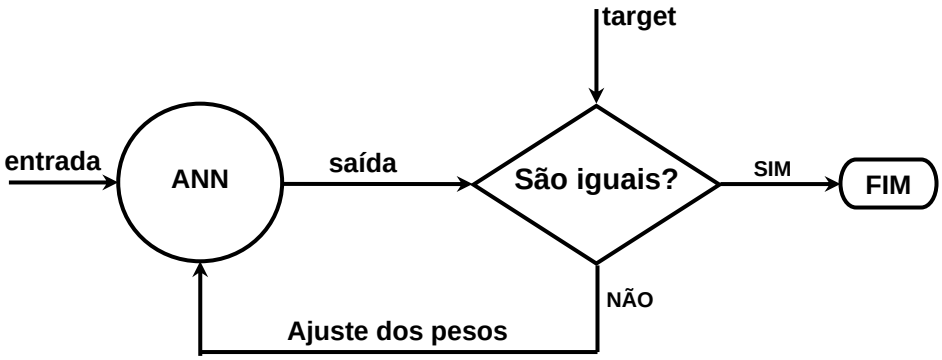

Figura 8 – Processo de treinamento de uma ANN (adaptado da *Neural Netwok Toolbox* do Matlab®)

Após o treinamento, a ANN aprende como deve proceder para outros dados de entrada no domínio do problema. Existem diversos algoritmos de treinamento ou aprendizado que diferem entre si principalmente pelo modo como os pesos são modificados. Quando um agente externo é utilizado para indicar à rede uma solução aceitável do problema, o aprendizado é dito ser supervisionado. Neste tipo de treinamento, tanto o vetor de entrada quanto o de saída é conhecido no problema. A ausência do agente externo conduz a um aprendizado não-supervisionado. No treinamento em modo incremental, a correção dos pesos é feita cada vez que uma entrada é apresentada à rede. Já no treinamento em modo *batch*, os pesos são corrigidos somente depois que todas as entradas são apresentadas (*Neural Netwok Toolbox* do Matlab®).

Uma rede neural *backpropagation* (BPN) usa a topologia *feedforward* e o algoritmo de aprendizado *backpropagation*. O algoritmo *backpropagation* realiza um aprendizado supervisionado onde as saídas desejadas são dadas como parte do vetor de treinamento. Na fase de treinamento, este algoritmo, opera em uma seqüência de dois passos. Primeiro, um sinal é apresentado à camada de entrada da rede e este sinal é propagado através da rede até que uma resposta seja produzida pela camada de saída. No segundo passo começa a fase de adaptação da rede onde a saída obtida pela rede é comparada à saída desejada para o sinal de entrada, produzindo um erro. Por fim, o erro é retropropagado através da rede para o ajuste dos pesos entre as camadas para produzir a saída correta (Bigus, 1996).

Uma BPN com apenas a camada de entrada e a camada de saída pode ser usada para construir um modelo de regressão linear, relacionando os múltiplos parâmetros de entrada às múltiplas saídas ou variáveis dependentes. O uso de apenas uma camada intermediária torna a

rede linear em uma não-linear, o que permite realizar uma regressão logística multivariada onde múltiplas saídas são possíveis de serem modeladas (Bigus, 1996).

3.2.3.1 O algoritmo Backpropagation

Considere a estrutura apresentada na Figura 9, o valor do *i*-ésimo neurônio da camada de entrada é passados para o *j*-ésimo neurônio da camada intermediária e o valor *net* do *j*-ésimo neurônio da camada intermediária vale conforme a equação (5)

$$net_j = \sum_{i=1}^{n} w_{ij} x_i + \theta_j \tag{5}$$

sendo x_i o *i*-ésimo valor de entrada, *n* o número de neurônios na camada de entrada, w_{ij} o peso da sinapse do neurônio *i* na camada de entrada para o neurônio *j* na camada intermediária, e θ_j o valor *threshold* (opcional e de valor igual a 1).

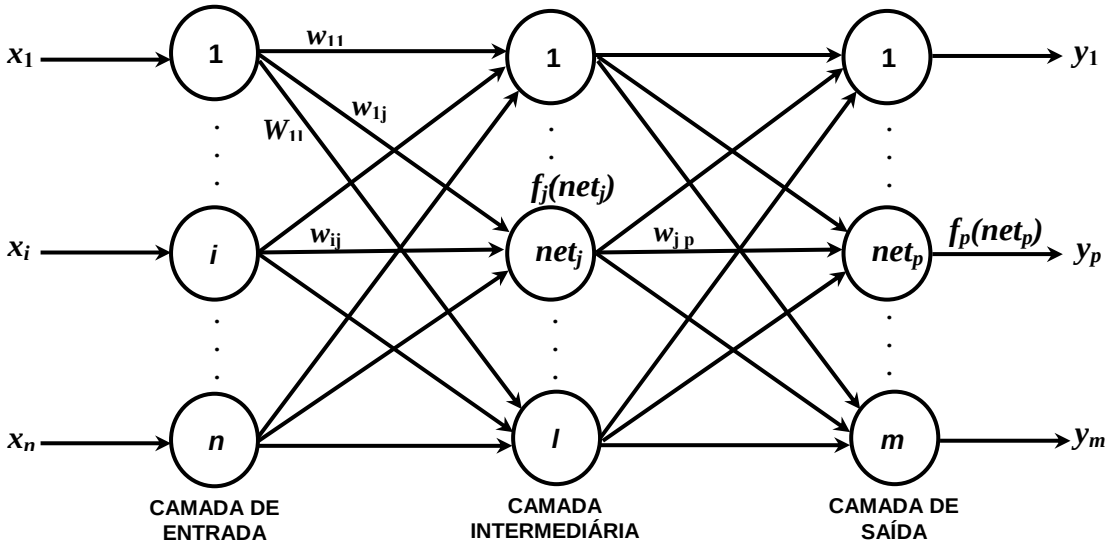

Figura 9 – Rede neural BPN (adaptado de Freeman & Skapura, 1991)

A saída da camada intermediária pode ser encontrada após a função de ativação $f_j\left(net_j\right)$ ser aplicada. Esta função pode ser a função sigmoidal apresentada na equação (6)

$$f\left(net\right) = \frac{1}{1+e^{-net}} \tag{6}$$

O sinal resultante da camada intermediária é propagado através da camada de saída. O valor *net* do *p*-ésimo neurônio da camada de saída da rede pode ser escrito como mostrado pela equação (7)

$$net_p = \sum_{j=1}^{l} w_{jp} f_j\left(net_j\right) + \theta_p \qquad (7)$$

onde θ_p é o valor *threshold* (opcional e de valor igual a 1), w_{jp} o peso da sinapse do neurônio *j* na camada intermediária para o neurônio *p* na camada de saída, *l* é o número de neurônios na camada intermediária. A saída da camada de saída da rede pode ser a função de ativação $f_p\left(net_p\right)$ do tipo linear.

No próximo passo do algoritmo, o sinal de saída y_p da rede é comparado com o sinal de saída z_p desejado (*target*), a diferença entre estes dois sinais é chamada de erro E_p pode ser calculado por $E_p = \left(z_p - y_p\right)$. O termo do erro de um neurônio na camada de saída pode ser calculado por $\delta_p = E_p f_p^{'}\left(net_p\right)$ e o termo do erro de um neurônio na camada intermediária pode ser calculado por $\delta_j = f_j^{'}\left(net_j\right)\sum_{p=1}^{m}\delta_p w_{jp}$. E para todos os neurônios de saída, um erro total pode ser formulado por meio do erro médio quadrático conforme a equação (8):

$$E = \frac{1}{2}\sum_{p=1}^{m}\delta_p{}^2 \,. \qquad (8)$$

Para encontrar o menor erro quadrático, a busca na superfície de erros se dá pelo gradiente negativo de *E* com relação aos pesos w_{jp}, $-\nabla E\left(w_{jp}\right)$, ou seja, os valores dos pesos podem ser ajustados tal que o erro total seja reduzido (Freeman & Skapura, 1991).

Os pesos dos neurônios na camada de saída são corrigidos de acordo com a equação (9)

$$w_{jp}\left(t+1\right) = w_{jp}\left(t\right) + \eta\delta_p f_j\left(net_j\right) \qquad (9)$$

sendo η a taxa de aprendizado que escala a mudança dos pesos na direção do gradiente. Este valor deve ser positivo e usualmente é menor do que 1.

Os erros na camada de saída são retropropagados para a camada intermediária para determinar as mudanças nos pesos dessa camada. A correção nos pesos dos neurônios da

camada intermediária depende de todos os termos de erro E_p na camada de saída e é dada conforme a equação (10):

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i . \quad (10)$$

3.2.4 Redes neurais artificiais via Matlab®

O método *steepest descent* é usado na BPN para minimizar a função erro. Neste método, a correção dos pesos e *thresholds* da rede são feitos na direção na qual a função de desempenho decresce mais rapidamente, ou seja, na direção oposta ao gradiente da função de desempenho. No software Matlab®, a função de treinamento em modo *batch* que utiliza o método *steepest descent* é a função *traingd*. Esta função permite configurar parâmetros, tais como:

- *epochs*: número de iterações do algoritmo de treinamento;
- *show*: exibe a condição do treinamento;
- *goal*: valor mínimo para a função de desempenho, ou seja, a tolerância no erro entre os valores encontrados pela rede e os valores desejados para a saída;
- *time*: tempo em segundos para realizar o treinamento;
- *min_grad*: menor valor para o gradiente;
- *lr*: representa a taxa de aprendizado, ou seja, o tamanho do passo para o ajuste do peso que é multiplicado ao gradiente. Um valor grande pode tornar o algoritmo instável, mas um valor muito pequeno demora na convergência do algoritmo.

Caso o algoritmo de aprendizado *backpropagation* fique preso em um mínimo local que não corresponda ao mínimo real do problema, um fator conhecido como momento pode ser empregado. A combinação deste fator momento com outras técnicas, tais como, *batching* e taxa de aprendizagem adaptativa, permite acelerar o aprendizado e torná-lo menos sensível a pequenas oscilações na superfície de erro. Na técnica do momento, um ajuste é adicionado às mudanças nos pesos, considerando uma fração de tempo τ. Assim, o ajuste do peso de um neurônio na camada intermediária pode ser realizado de acordo com a equação (11):

$$\Delta w_{ij} = \eta \delta_j x_i + \tau . \Delta w_{ij} \quad (11)$$

e o ajuste do peso de um neurônio na camada de saída está representado na equação (12)

$$\Delta w_{jp} = \eta \delta_p f_j \left(net_j \right) + \tau . \Delta w_{jp} \tag{12}$$

A função de treinamento do Matlab® em modo *batch* que utiliza o método *steepest descent*, considerando o fator momento, é a função *traingdm*. Outra função, que além de utiliza o fator momento, utiliza uma taxa de aprendizado adaptativa é a função *traingdx*. Quando esta taxa de aprendizado possui um valor muito grande, não há a garantia de o valor mínimo desejado ser encontrado.

Como já mencionado, uma BPN é uma rede neural que utiliza a topologia *feedforward* e o algoritmo de aprendizado *backpropagation*. O processo de treinamento de uma rede neural *feedforward* pode ser feito por meio dos seguintes passos:

1. ***Dados de entrada***: escolha dos dados de entrada e de saída da rede neural que serão utilizados no treinamento da rede;
2. ***Construção da rede***: criação do objeto rede e configuração dos parâmetros desejados;
3. ***Treinamento***: fase de treinamento para o ajuste dos pesos e valores limites (*thresholds*) entre as camadas da rede, conduzindo os valores de saída próximos aos valores desejados (*target*);
4. ***Simulação***: fase de simulação ou teste da rede para novos dados de entrada.

Capítulo 4

DETECÇÃO DE DANOS

4.1 INTRODUÇÃO

Muitas estruturas, durante sua vida útil, são submetidas a diversos tipos e formas de carregamentos estáticos e dinâmicos. Estes carregamentos associados ao processo de deterioração estrutural podem provocar diferentes tipos de danos nas estruturas. A caracterização do dano e o conhecimento de quais mudanças nas propriedades dos materiais correspondem aos danos, dependem do tipo de material e da configuração estrutural. Por exemplo, em materiais metálicos os danos mais comuns são as trincas e a corrosão, enquanto que em materiais compósitos, delaminações e danos provocados por impacto são mais comuns (Cecchini, 2005).

A localização e a quantificação do dano consistem de um problema de identificação de sistemas ou um problema inverso. Muitas técnicas têm sido usadas para localizar e identificar danos em uma estrutura e diversos algoritmos tem sido implementados para caracterizar estes danos. Os algoritmos genéticos (GA's), as redes neurais artificiais (ANN's) e o filtro de Kalman (KF) podem ser utilizados na solução do problema inverso. O problema direto consiste em determinar os efeitos provocados por dadas informações conhecidas que são aplicadas na estrutura por meio de um processo matemático. Dentre as informações disponíveis para o problema direto, os carregamentos e os defeitos presentes nas estruturas podem ser citados.

Neste trabalho, o método de elementos de contorno (BEM) para o potencial foi utilizado para a resolução do problema direto de detecção de danos, ou seja, a modelagem térmica do problema de detecção de furos circulares na placa é feita por meio do BEM para o potencial e utilizando elementos constantes. O problema a ser modelado é um problema de transferência de calor por condução simples, considerando a distribuição de temperatura sobre a superfície de uma placa fina. Possíveis furos presentes na estrutura são considerados adiabáticos. Simulações por elementos de contorno fornecem as informações necessárias para o problema inverso, levando em consideração o tipo de carregamento aplicado, a localização e o raio do furo. O problema inverso pode ser modelado como um problema de otimização ou como um problema de identificação de parâmetros. Na otimização, a minimização da diferença entre os valores medidos (simulados) da diferença de potencial (entre a placa sem dano e a placa com dano) e os calculados pelo programa é desejada. Como esta diferença é, em geral, não-convexa, os GA's são utilizados para resolver o problema de otimização global. Na identificação de parâmetros, ANN's e o KF podem ser utilizados.

Resumindo, a detecção de danos em estruturas, dentre outros objetivos, tenta:

- detectar a presença de defeitos, como furos circulares em placas;
- identificar os parâmetros dos danos, como a localização e o raio de furos circulares.

4.2 MÉTODO DE ELEMENTOS DE CONTORNO PARA O PROBLEMA DE POTENCIAL

O programa usando o BEM para a resolução do problema direto de detecção de danos foi escrito em Matlab® tendo por base os trabalhos desenvolvidos em Fortran® por Brebbia & Dominguez (1992). Neste trabalho, uma função chamada POCONBE foi utilizada. Esta função faz o cálculo do potencial nos pontos internos da placa utilizando elementos constantes. A listagem completa do programa de resolução do problema direto está apresentada no Anexo A.

O fluxograma do programa desenvolvido está apresentado na Figura 10. O programa principal chama a função INPUTPC responsável pela entrada de dados. É nesta rotina onde estão definidos o número de discretizações da placa, as condições de contorno e o número de

pontos internos. O conjunto de chamadas das funções CPLOT, GHMATPC, APLYBC, REORDER, INTERPC, PRINT e outras manipulações matemáticas necessárias ao programa formam o programa POCONBE. A função CPLOT apresenta uma figura com o contorno da placa discretizado e com os pontos internos nos locais determinados. A função GHMATPC calcula as matrizes H e G do BEM e a função APLYBC aplica as condições de contorno do problema. Tendo as matrizes G e H já com as condições de contorno aplicadas, o sistema é resolvido e em seguida, os vetores de potencial e fluxo são reordenados pela função REORDER para a impressão na tela. Por fim, o potencial nos pontos internos é calculado pela função INTERPC e os resultados podem ser impressos pela chamada da função PRINT. Tendo encontrado o potencial dos pontos internos, a diferença de potencial entre a placa sem furo e a placa com furo é calculada e o resultado obtido é normalizado. Os resultados obtidos são armazenados para que um pós-processamento possa ser feito.

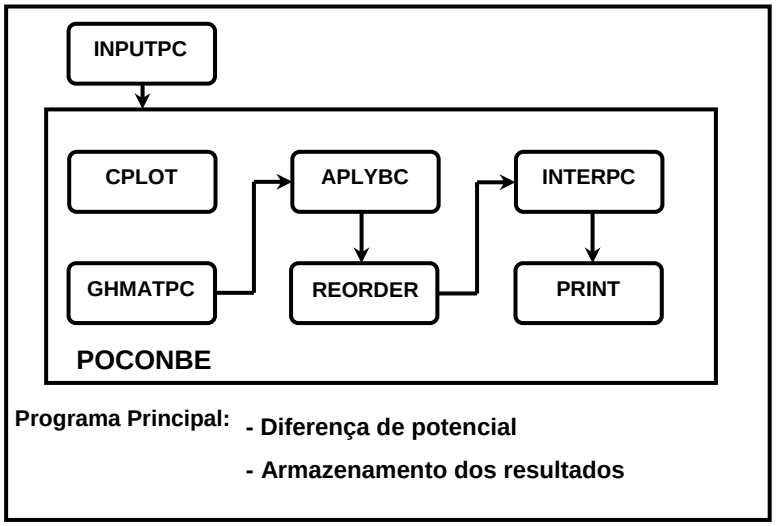

Figura 10 – Fluxograma do BEM para o potencial

Uma observação deve ser feita com respeito à discretização da placa e do furo. Como a placa apresenta um domínio fechado, a numeração dos elementos no BEM é feita no sentido anti-horário. Já o furo sendo um domínio aberto, a numeração é feita no sentido horário. Veja as Figuras 11(a) e 11(b).

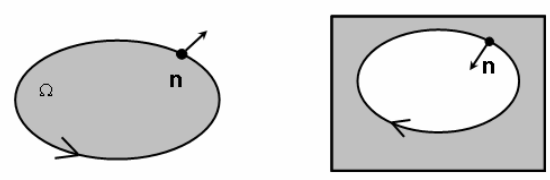

Figura 11 – Numeração dos elementos no BEM: (a) domínio fechado; (b) domínio aberto.

4.3 O PROBLEMA DE DETECÇÃO DE DANOS

Para determinar os parâmetros desconhecidos do dano por meio do GA, um funcional pode ser definido como a diferença entre os valores medidos (simulados) da diferença de potencial (entre a placa sem dano e a placa com dano) e os calculados pelo programa de detecção de danos. Os valores de potencial são simulados por meio do BEM para o potencial em 49 pontos internos da placa, conforme mostrado na Figura 12. A formulação do funcional está mostrada na equação (13)

$$\mathbf{J}_j = \frac{1}{2}\sum_{i=1}^{n}\left(\mathbf{medido}_i - \mathbf{calculado}_{ji}\right)^2 \qquad (13)$$

onde:

n – número de sensores presentes na placa;

$\mathbf{medido}_j$ - vetor linha de valores simulados da diferença de potencial, representando os valores medidos na placa para um determinado dano;

$\mathbf{calculado}_{ji}$ - vetor de valores calculados pelo programa de detecção de danos para cada indivíduo j.

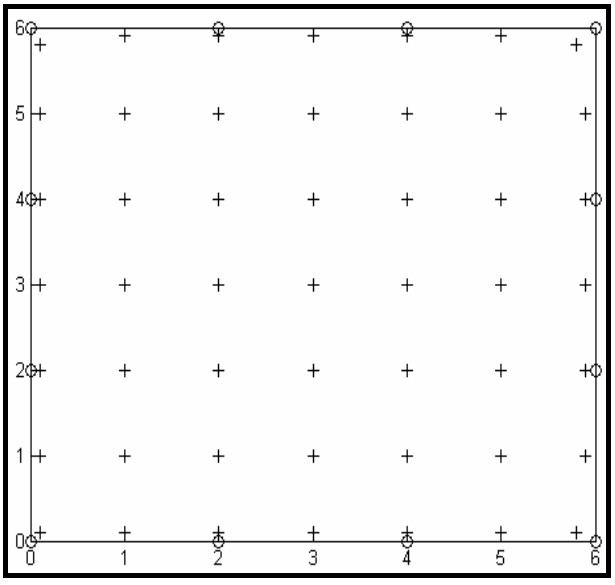

Figura 12 – Distribuição dos sensores para o GA

Este funcional corresponde à função de avaliação do GA e a sua minimização permite que o programa de detecção de danos encontre os parâmetros desconhecidos do dano. Muitos trabalhos vem sendo desenvolvidos para a detecção de danos utilizando o BEM e técnicas de

otimização para a minimização do funcional. Dentre os trabalhos existentes, os trabalhos desenvolvidos por Stravoulakis & Antes (1998) e Burczynski & Beluch (2001) podem ser citados.

Nas ANN's, informações a respeito da diferença de potencial na placa são fornecidas na entrada da rede e os dados, quantidade, localização e raio do furo, são fornecidos na saída. Furos de diferentes tamanhos e em diferentes locais podem fazer parte dos dados fornecidos à rede. Tendo definido os dados de entrada e de saída da rede, o próximo passo é construir a rede para então treiná-la. Por fim, a rede pode ser testada para outras informações de diferença de potencial, obtendo como resposta a quantidade, a localização e o tamanho do furo. Para a montagem dos dados de entrada da rede, inicialmente 25 pontos internos, representando os sensores, foram considerados. Em seguida, o número de sensores foi diminuído para 15, 9 e 5, respectivamente. No presente trabalho, os sensores foram uniformemente distribuídos na placa e nenhum estudo a respeito do seu posicionamento foi realizado. A distribuição dos sensores na placa, para cada caso, está mostrada nas Figuras 13(a), 13(b), 13(c) e 13(d).

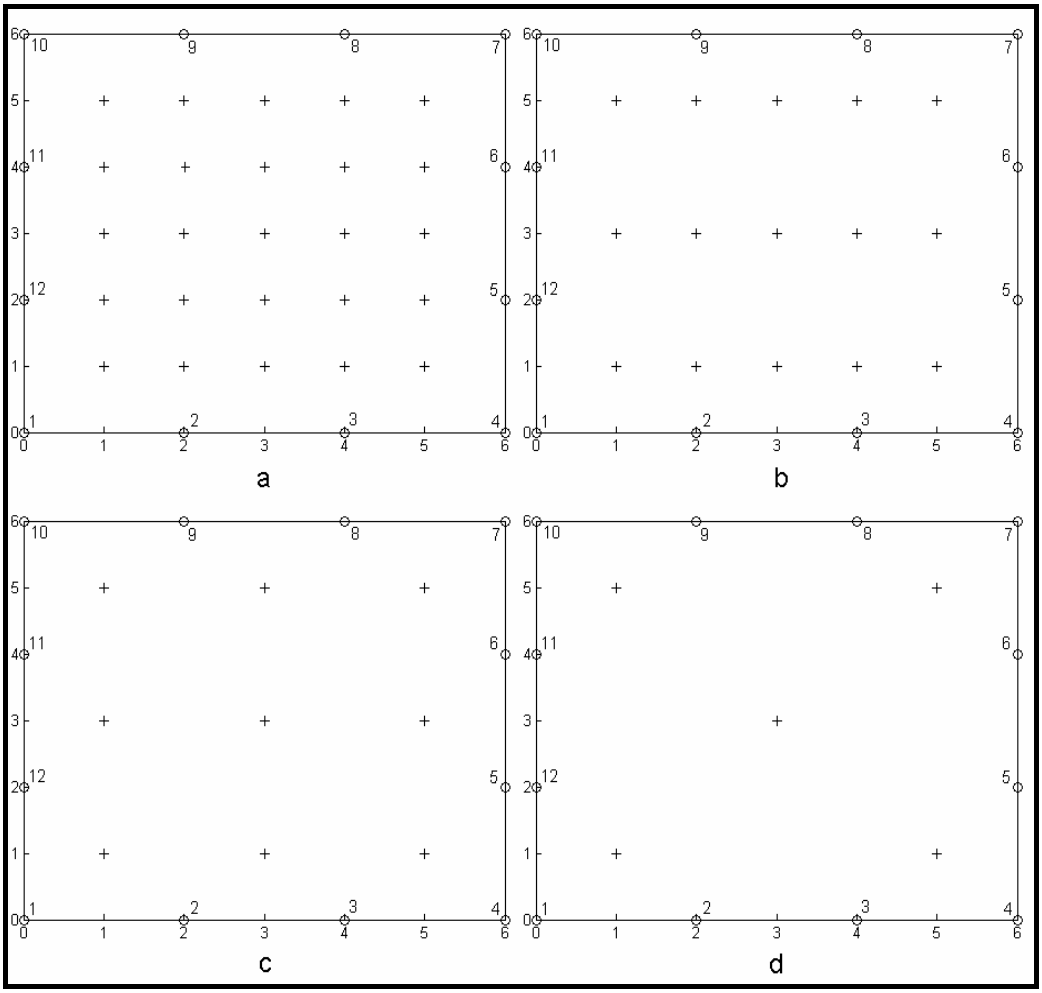

Figura 13 – Distribuição dos sensores para a ANN: (a) 25, (b) 15, (c) 9, (d) 5

4.3.1 Configuração do cromossomo do algoritmo genético

Para resolver o problema de detecção de danos, uma população inicial é fornecida ao GA. Esta população inicial é formada por indivíduos que constituem uma possível solução do problema. Para determinar a população inicial, o cromossomo que representa um indivíduo da população pode ser formado conforme o vetor apresentado na equação (14):

$$c = [g_1 \quad g_2 \quad g_3 \quad g_4 \quad \dots \quad g_{n+3}] \tag{14}$$

onde:

g_1 – primeiro gene representando a posição *x* do centro do furo;

g_2 – segundo gene representando a posição *y* do centro do furo;

g_3 – terceiro gene representando o raio do furo;

$g_4 \quad \dots \quad g_{n+3}$ – quarto gene em diante representando as medidas da diferença de potencial entre a placa sem furo e a placa com furo.

Quando a placa apresenta dois furos, a codificação para o cromossomo obedece a seguinte estrutura: o gene 1 representa o número de furos na placa, os genes 2 e 3 representam as coordenadas do primeiro furo, os genes 4 e 5 representam as coordenadas do segundo furo, os genes 6 e 7 representam o raio do primeiro e do segundo furo, respectivamente, e os genes a partir do gene 8 representam as medidas da diferença de potencial na placa sob estudo. Essa nova configuração pode ser utilizada para a codificação da presença de um furo apenas, observando que as coordenadas do centro e o raio para o segundo furo são configurados com o valor nulo. A listagem completa do programa de detecção de danos utilizando o GA está apresentada no Anexo B.

4.3.2 Configuração dos dados de entrada da rede neural artificial

As informações em nove posições diferentes de furos na placa são obtidas por meio do BEM para o problema do potencial, formando o primeiro conjunto de dados. Admitindo um furo com um raio igual a 0,15 *cm* em cada uma das posições (0,5;0,5) *cm*, (0,5;3) *cm*, (0,5;5,5) *cm*, (3;0,5) *cm*, (3;3) *cm*, (3;5,5) *cm*, (5,5;0,5) *cm*, (5,5;3) *cm* e (5,5;5,5) *cm* por vez, os valores normalizados da diferença entre os potenciais da placa sem furo e com furo foram encontrados. A Figura 14 mostra cada possível localização dos furos na placa, considerando

25 sensores. Por fim, os valores da diferença normalizada de potencial, o número de furos, a posição *x*, a posição *y* e o raio do furo foram armazenados para um posterior pós-processamento pela ANN.

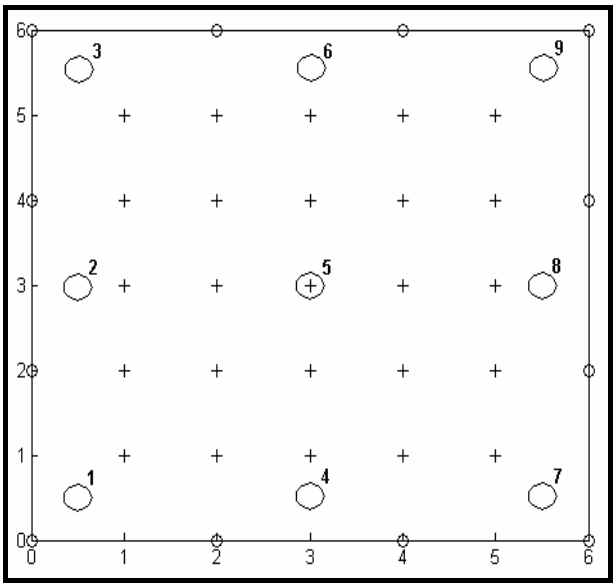

Figura 14 – Possíveis localizações dos furos na placa com 25 sensores

O esquema anterior de montagem de dados é seguido quando a detecção de mais de um furo na placa é realizada. Caso o número de furos na placa seja dois, os dados armazenados são os valores da diferença de potencial normalizada na placa, o número de furos, os parâmetros do primeiro furo e os parâmetros do segundo furo. No programa que detecta até dois furos, é possível detectar apenas um furo na estrutura, observando que os locais onde supostamente haveria os parâmetros do segundo furo possuem valores nulos na configuração do cromossomo. A listagem completa dos programas de detecção de danos utilizando a ANN, considerando um furo apenas, e que detecta até dois furos, está apresentada no Anexo C.

Para tornar o programa de detecção de danos o mais genérico possível, informações a respeito do número e parâmetros dos furos devem ser generalizadas. Isto possibilita o programa detectar mais de um furo na estrutura se houver e não até dois furos como foi estudado. Antes de tornar o programa mais genérico, um melhoramento na coleta dos dados de entrada deve ser feita. Outros métodos de elementos de contorno e outros carregamentos aplicados na estrutura podem ser necessários para fornecer mais informação a respeito do dano.

Capítulo 5

RESULTADOS E DISCUSSÕES

5.1 ANÁLISE DOS RESULTADOS OBTIDOS PELO ALGORITMO GENÉTICO

Neste tópico são analisados os resultados obtidos pelo programa desenvolvido de detecção de danos para um problema de transferência de calor em uma placa fina. Inicialmente uma placa sem dano e com as dimensões (6×6) *cm* conforme mostrada na Figura 15 foi simulada por meio do método de elementos de contorno (BEM). Esta placa foi discretizada por 12 elementos de contorno e o valor do potencial em 49 pontos internos foi encontrado. As condições de contorno para o problema estão representadas nesta figura, onde *q* representa o fluxo de calor e *u* a temperatura no contorno.

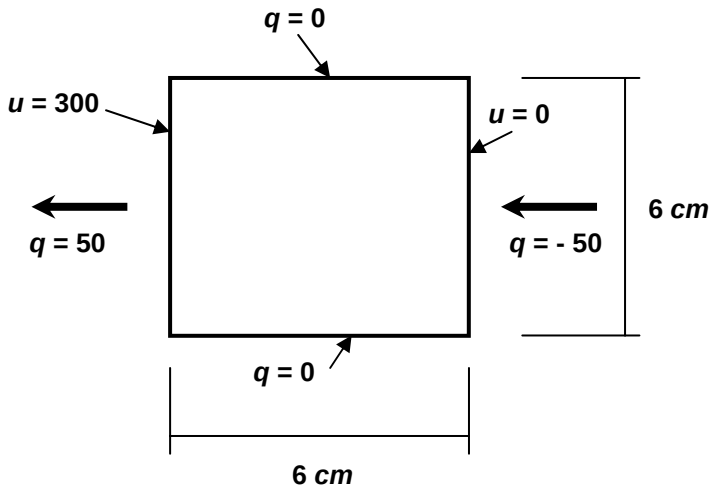

Figura 15 – Condições de contorno de uma placa quadrada (Brebbia & Dominguez, 1992)

Depois, uma placa com um furo central de raio 0,06 *cm*, com as mesmas dimensões e condições de contorno também foi simulada, e os resultados encontrados para o potencial foram comparados com a placa sem dano. A diferença entre os valores de potencial da placa sem dano com a placa com dano foi normalizada e esta diferença pode ser visualizada na Figura 16.

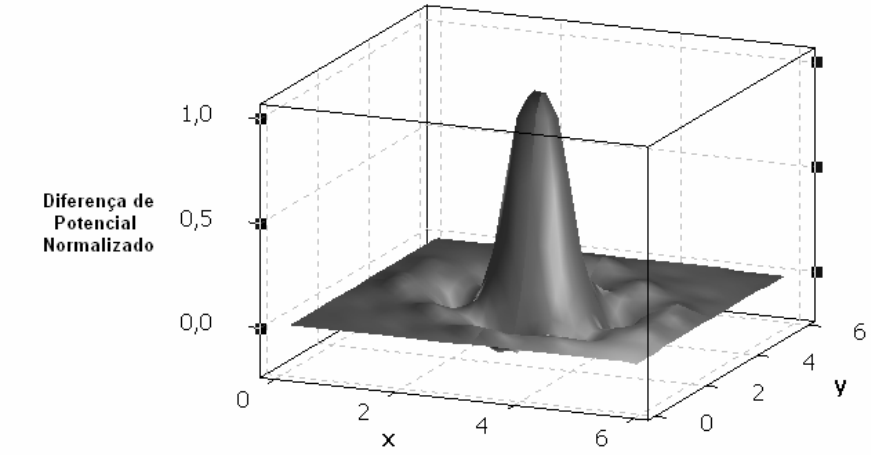

Figura 16 – Diferença de potencial normalizado para uma placa com furo central

Os resultados apresentados nesta seção foram obtidos em um computador PENTIUM IV de 3 GHz rodando a aplicação para o algoritmo genético (GA) escrito no Matlab®. O critério de parada do GA foi configurado para o número de gerações igual a 75. Os demais parâmetros do GA foram configurados como:

- ✓ Tamanho da população inicial: 330 indivíduos;
- ✓ Probabilidade de cruzamento: 0,8;

- ✓ Probabilidade de mutação: 0,2;
- ✓ Elitismo: 2;
- ✓ Função de cruzamento: foi considerada a *crossoverheuristic* com uma taxa igual a 1,3;
- ✓ Função de mutação: foi considerada a *mutationgaussian*.

Para o programa que detecta apenas um furo, a configuração do cromossomo apresentada no Capítulo 4 foi seguida para formar a população inicial do GA. Por meio do BEM para o potencial, os valores de potencial da placa sem dano e da placa com dano foram obtidos. Inicialmente para um furo apenas, os raios considerados no problema foram 0,15 *cm*, 0,03 *cm* e 0,09 *cm*. Para cada um destes raios, a coordenada *x* do centro do furo foi variada de 0,5 *cm* a 5 *cm* e a coordenada *y* do centro do furo foi variada de 0,5 *cm* a 5,5 *cm*, ambas as coordenadas com um passo de 0,5 *cm*. Então, 110 posições diferentes para cada raio na placa foram simuladas e os respectivos valores da diferença de potencial foram armazenados para efetuar o pós-processamento. No programa que executa o GA, os valores da diferença de potencial foram normalizados, levando em consideração o maior valor desta diferença. Finalmente, a população inicial com 330 indivíduos pode ser formada. Como os valores de potencial próximos à borda direita (temperatura igual a zero) da placa são próximos a zero, a diferença de potencial é utilizada ao invés do uso direto do valor de potencial.

Para o programa que detecta até dois furos, a formação da população inicial levou em consideração indivíduos representando um furo apenas e outros indivíduos representando dois furos. Para um furo, os raios 0,15 *cm*, 0,03 *cm* e 0,09 *cm* foram novamente considerados, formando 330 indivíduos. Para dois furos, enquanto um furo de raio 0,03 *cm* na posição (3;3) *cm* era mantido fixo, um furo de raio 0,15 *cm* varria a placa (a coordenada *x* do centro do furo foi variada de 0,5 *cm* a 5 *cm* e a coordenada *y* do centro do furo foi variada de 0,5 *cm* a 5,5 *cm*, ambas as coordenadas com um passo de 0,5 *cm*). Para o segundo grupo de indivíduos, os valores de raio foram invertidos, ou seja, na posição fixa (3;3) *cm* foi considerado um furo de raio 0,15 *cm* e para a posição que variava, um furo de raio 0,03 *cm*. Para o terceiro grupo de indivíduos formados, foram considerados ambos os furos com um raio de 0,03 *cm*. Para completar a formação da população inicial do GA, os valores da diferença de potencial foram normalizados com relação ao maior valor da diferença de potencial. Finalmente, a formação do cromossomo foi feita conforme explicado no Capítulo 4. Como a população de indivíduos formada possui tanto indivíduos com um furo e indivíduos com dois furos, os vetores representando indivíduos com um furo devem possuir o mesmo comprimento dos vetores representando indivíduos com dois furos. Então, quando um indivíduo com um furo apenas é

considerado, as informações relativas ao segundo furo devem ser consideradas nulas no cromossomo para a formação do indivíduo.

A Figura 17 mostra a discretização da placa e o resultado encontrado pelo programa de detecção de danos para um furo na posição (3;3) *cm* e raio igual a 0,06 *cm*. O programa foi executado apenas 5 vezes, pois não houve diferença significativa quando este valor era aumentado, e o tempo médio de execução foi de 100,3844 *s*. Os resultados encontrados com 99,7% de confiança (média mais ou menos três vezes o desvio padrão) estão apresentados na equação (15):

$$\begin{aligned} xc &= (3,014 \pm 0,074)\ cm \\ yc &= (2,971 \pm 0,091)\ cm \\ r &= (0,042 \pm 0,021)\ cm \\ J &= (0,000 \pm 0,000)\ ^{\circ}C \end{aligned} \tag{15}$$

onde:

xc - coordenada x do furo;

yc - coordenada y do furo;

r - raio do furo;

J - melhor valor encontrado para o funcional.

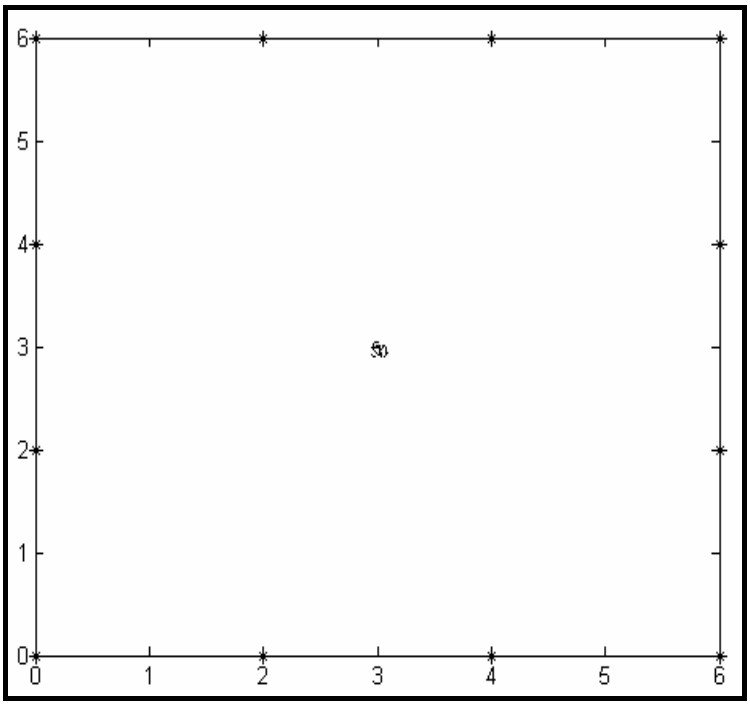

Figura 17 – Discretização da placa e região da presença do furo

A Figura 18 mostra a região do furo em uma escala ampliada. A posição real do furo está representada em linha contínua e os resultados encontrados pelo programa de detecção de danos em pontilhado.

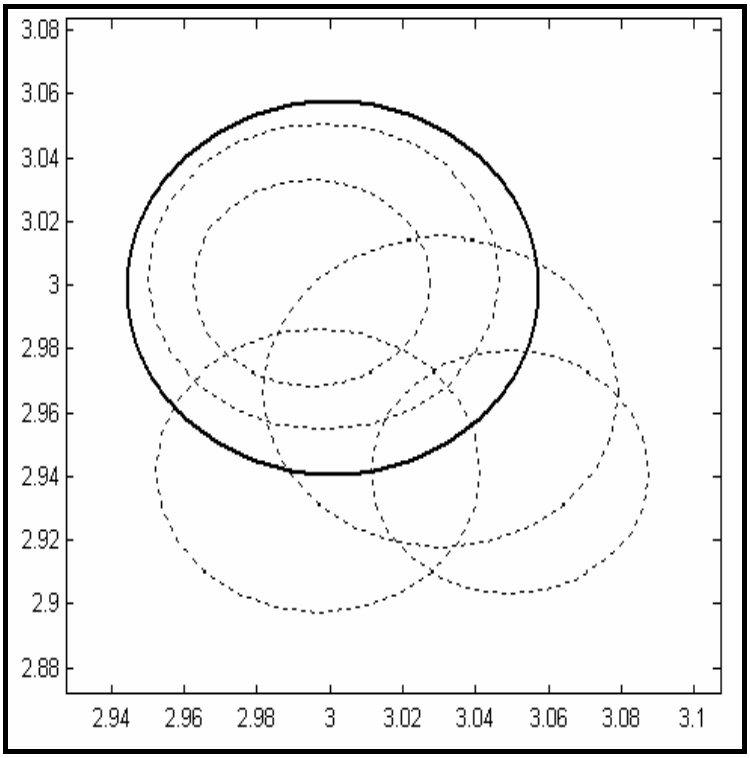

Figura 18 – Região da presença furo em 5 rodadas

Pelos resultados obtidos é verificado que o programa desenvolvido não consegue encontrar uma resposta exata, mas a região de ocorrência do dano. Uma possível causa seria porque no GA há uma pequena presença de mutação e uma função de cruzamento que não é igual para cada execução do algoritmo, ou seja, há uma probabilidade de ocorrência associada.

Mudando o valor de elitismo de 2 para 10, ou seja, garantindo que 10 indivíduos sobrevivam na próxima geração, e mantendo os demais parâmetros na configuração anterior, os resultados encontrados foram os apresentados na equação (16). O tempo médio de execução foi de 95,2282 *s*.

$$\begin{aligned} xc &= (3{,}000 \pm 0{,}005)\ cm \\ yc &= (3{,}001 \pm 0{,}010)\ cm \\ r &= (0{,}061 \pm 0{,}038)\ cm \\ J &= (0{,}000 \pm 0{,}000)\ ^{\circ}C \end{aligned} \tag{16}$$

A nova região do furo em escala ampliada está apresentada na Figura 19. Nesta figura é possível verificar que os furos são concêntricos, o que gerou a um pequeno valor de incerteza na posição do furo. O raio ainda continua pouco sensível à variação dos parâmetros do GA, apresentando uma incerteza significativa.

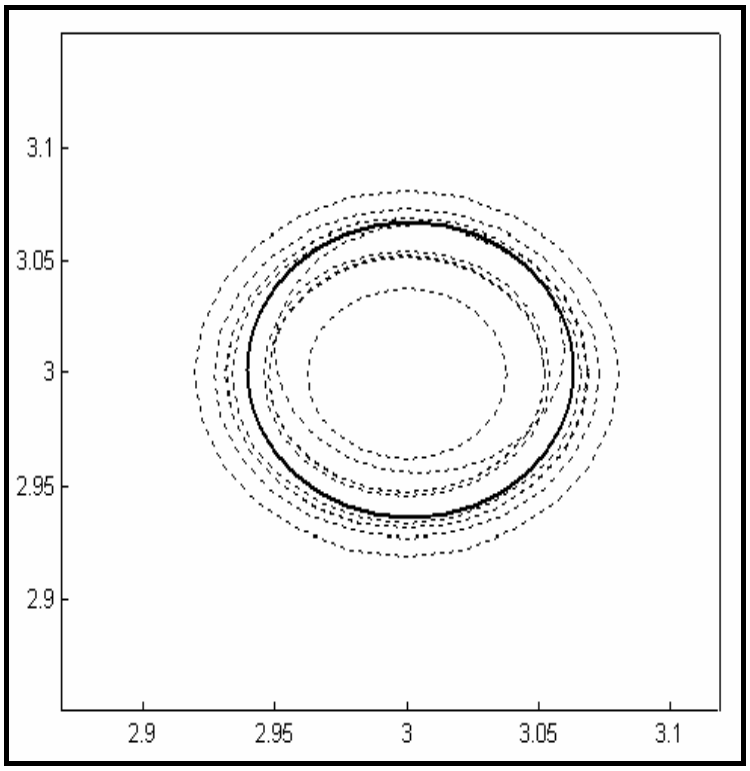

Figura 19 – Região da presença furo para a nova configuração

Muitas outras configurações dos parâmetros do GA foram testadas e outras posições do furo na placa foram analisadas, sendo a melhor configuração a apresentada primeiramente. Então, considerando um furo na posição (4;5) *cm* e raio igual a 0,06 *cm*. O programa foi executado 5 vezes e o tempo médio de execução foi de 119,8032 *s*. Os resultados encontrados com 99,7% de confiança estão apresentados na equação (17)

$$\begin{aligned} xc &= (3,965 \pm 0,124)\ cm \\ yc &= (5,002 \pm 0,089)\ cm \\ r &= (0,006 \pm 0,021)\ cm \\ J &= (0,000 \pm 0,000)\ ^{\circ}C \end{aligned} \tag{17}$$

A Figura 20 mostra a região de ocorrência do furo e a discretização da placa. A Figura 21 mostra a região do furo em uma escala ampliada. Como já foi dito anteriormente, a posição

real do furo está representada em linha contínua e os resultados encontrados pelo programa de detecção de danos em pontilhado.

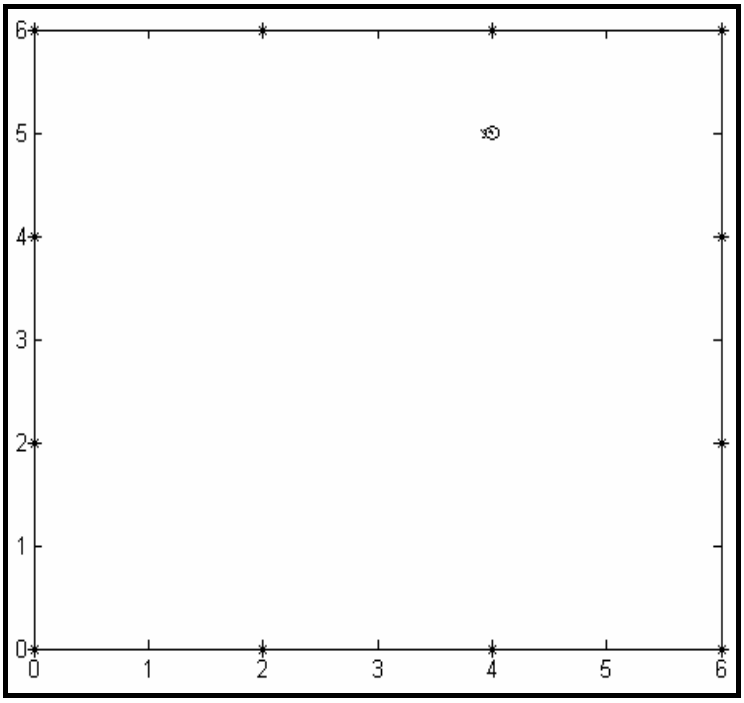

Figura 20 – Discretização da placa e região da presença do furo na nova posição

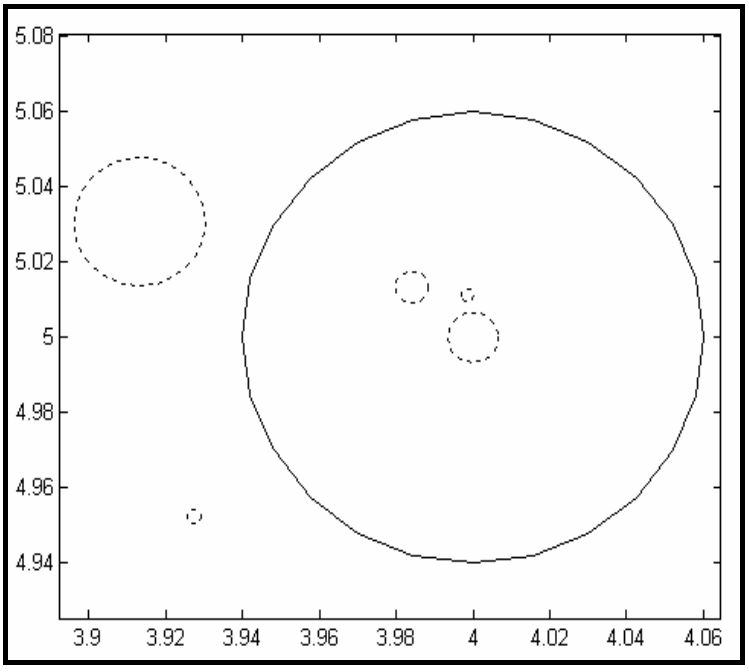

Figura 21 – Região da presença furo para a nova posição

Pelos resultados dados na equação (17) e pela Figura 20 pode ser verificado que o programa encontra a região de ocorrência do furo, porém o programa tem dificuldade mais uma vez de encontrar o raio exato do furo. Uma possível justificativa para este problema é

que existe pouca informação fornecida ao GA, ou seja, é necessário melhorar a coleta de dados do problema direto de detecção de danos.

O problema de detecção de danos pode ser formulado para detectar a presença de mais de um furo na estrutura, observando que a codificação do GA deve ser alterada como já discutido no Capítulo 4. No programa que detecta até dois furos, as modificações feitas nos parâmetros do GA foram considerar uma população inicial com 660 indivíduos, e a função de cruzamento *crossoverheuristic* com uma taxa igual a 1,25. Este programa foi utilizado para detectar a presença de apenas um furo na posição (3;3) *cm* e raio igual a 0,06 *cm*. O tempo médio de execução do programa em 10 rodadas do GA foi de 78.9813 *s*. A equação (18) mostra o numero de furos n detectados pelo programa e os resultados encontrados pelo programa para a localização dos mesmos.

$$\begin{aligned} &n = 1 \\ &xc_1 = \left(3,006 \pm 0,088\right)\ cm \\ &yc_1 = \left(3,002 \pm 0,104\right)\ cm \\ &r_1 = \left(0,046 \pm 0,031\right)\ cm \\ &xc_2 = \left(0,000 \pm 0,000\right)\ cm \\ &yc_2 = \left(0,000 \pm 0,000\right)\ cm \\ &r_2 = \left(0,000 \pm 0,000\right)\ cm \end{aligned} \tag{18}$$

Pelos resultados apresentados na equação (18) é possível verificar que o programa detecta corretamente a presença de apenas um furo na placa. Os resultados apresentados são similares aos obtidos anteriormente pelo programa que detecta apenas um furo.

O programa que detecta dois furos foi executado 10 vezes para uma placa com um furo na posição (1;1) *cm* e raio de 0,03 *cm* e outro na posição (3,5;3,5) *cm* e raio de 0,09 *cm*. Apesar de o programa convergir, ele não detectou a ocorrência dos dois furos, apenas a presença de um furo foi observada, conforme mostrado na equação (19). Isto se deve, em parte, ao fato dos valores dos raios serem muito pequenos e um deles estar próximo à borda da placa.

$$
\begin{aligned}
&n = 1 \\
&xc_1 = (1{,}107 \pm 0{,}161)\ cm \\
&yc_1 = (1{,}249 \pm 0{,}087)\ cm \\
&r_1 = (0{,}118 \pm 0{,}058)\ cm \\
&J = (0{,}000 \pm 0{,}000)\ ^{\circ}C
\end{aligned}
\tag{19}
$$

Novamente o programa que detecta dois furos foi executado, considerando uma placa com um furo na posição (1;3) *cm* e raio de 0,03 *cm* e outro na posição (5;3) *cm* e raio de 0,09 *cm*. Os resultados obtidos estão apresentados na equação (20)

$$
\begin{aligned}
&n = 2 \\
&xc_1 = (1{,}336 \pm 0{,}956)\ cm \\
&yc_1 = (2{,}960 \pm 0{,}791)\ cm \\
&r_1 = (0{,}03 \pm 0{,}109)\ cm \\
&xc_2 = (2{,}882 \pm 0{,}468)\ cm \\
&yc_2 = (2{,}882 \pm 0{,}468)\ cm \\
&r_2 = (0{,}184 \pm 0{,}101)\ cm \\
&J = (0{,}000 \pm 0{,}000)\ ^{\circ}C
\end{aligned}
\tag{20}
$$

Pelos dois exemplos anteriores, verifica-se que o programa encontra a região de ocorrência de um dos furos e tem dificuldade em encontrar a região do segundo furo. A falta de informações consistentes fornecidas pelo BEM tem grande peso neste problema. A codificação do cromossomo do GA tem que ser a mais aleatória para que considere todas as possíveis soluções do problema.

Quando o cromossomo do GA foi codificado, determinadas localizações e raios foram previamente escolhidos para formar a população inicial. Talvez mais informações devessem ser fornecidas ao GA para formar esta população inicial e evitar que o programa de detecção de danos convergisse para um valor mínimo que não fosse o real. Os resultados obtidos para uma placa com um furo na posição (1;3) *cm* e raio de 0,03 *cm* e outro na posição (3;3) *cm* e raio de 0,15 *cm* estão apresentados na equação (21). A Figura 22 apresenta a região de ocorrência dos dois furos em 5 rodadas do programa de detecção de danos. Esta simulação considera os dados que pertencem à população inicial. A incerteza presente nos resultados é devida ao fato de haver uma pequena mutação no GA e à própria função de cruzamento deste algoritmo.

$$
\begin{aligned}
&n = 2 \\
&xc_1 = (1{,}193 \pm 0{,}732)\ cm \\
&yc_1 = (2{,}967 \pm 0{,}483)\ cm \\
&r_1 = (0{,}034 \pm 0{,}181)\ cm \\
&xc_2 = (2{,}990 \pm 0{,}252)\ cm \\
&yc_2 = (2{,}990 \pm 0{,}252)\ cm \\
&r_2 = (0{,}180 \pm 0{,}033)\ cm \\
&J = (0{,}000 \pm 0{,}000)\ ^{\circ}C
\end{aligned}
\tag{21}
$$

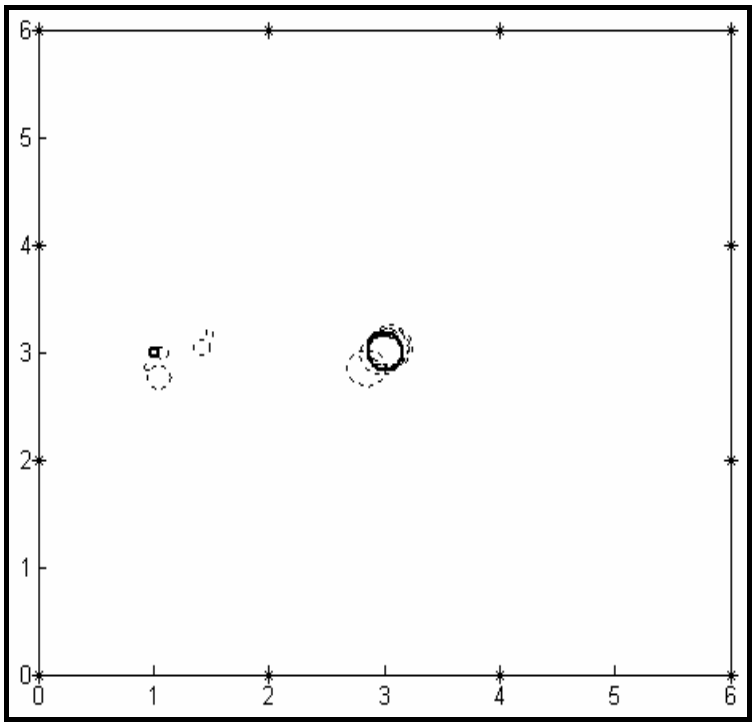

Figura 22 – Região de ocorrência dos dois furos encontrada pelo programa

A detecção de danos por meio do GA exige a configuração do cromossomo para cada tipo de dano e a decisão de qual a melhor configuração dos seus parâmetros para um determinado problema é uma tarefa árdua. Além disso, inúmeras avaliações da função *fitness* são feitas neste algoritmo, o que provoca um alto custo computacional. Sendo assim, emprega-se a técnica de redes neurais artificiais (ANN's) para a detecção do dano na estrutura em tempo real.

5.2 ANÁLISE DOS RESULTADOS OBTIDOS PELA REDE NEURAL ARTIFICIAL

Considerando o mesmo problema de transferência de calor do item anterior, inicialmente foi estudada a presença de apenas um furo na estrutura. Em seguida, foi verificada a influência nos resultados quando o número de sensores na placa é diminuído.

Para o primeiro conjunto de dados fornecidos à rede, 9 possíveis posições do furo e com um raio de 0,15 *cm* foram consideradas conforme apresentado no Capítulo 4. Numa tentativa de melhorar a qualidade dos resultados obtidos pela rede, os dados de entrada foram aumentados e as 9 posições diferentes de furos foram consideradas com os raios 0,15 *cm* e 0,05 *cm*. As informações foram coletadas considerando a distribuição de sensores apresentada no Capítulo 4.

Assim como os dados de entrada, os valores utilizados para testar a rede após o treino da mesma foram montados seguindo o esquema de distribuição dos sensores na placa apresentado no Capítulo 4. Um furo de raio 0,10 *cm* em cada uma das posições dadas ((1;1) *cm*, (2;4) *cm*, (3;3) *cm*, (4;2) *cm* e (5;5) *cm*) foi considerado para testar a rede. Então, para o primeiro conjunto de dados de entrada da rede (furos com um raio de 0,15 *cm*), os resultados obtidos após o treinamento da ANN podem ser visualizados na Figura 23, considerando 25 sensores e estão sumarizados na Tabela 1. A melhor escolha para os parâmetros da rede neural *backpropagation* (BPN) foi:

- ✓ Número de neurônios na camada de entrada: 50
- ✓ Número de neurônios na camada intermediária: 8
- ✓ Número de neurônios na camada de saída: 4 (porque são 4 parâmetros de saída: número de furos, posição x, posição y e raio)
- ✓ Função de ativação da camada de entrada: tangente hiperbólica sigmoidal (*tansig*)
- ✓ Função de ativação da camadaintermediária: tangente hiperbólica sigmoidal (*tansig*)
- ✓ Função de ativação da camada de saída: linear (*purelin*)
- ✓ Função de treinamento: *gradient descent* com momento e taxa de aprendizagem adaptativa (*traingdx*)
- ✓ Erro desejado: 1.10^{-5}
- ✓ Número de iterações: 10.000
- ✓ Taxa de aprendizado: 0.0005

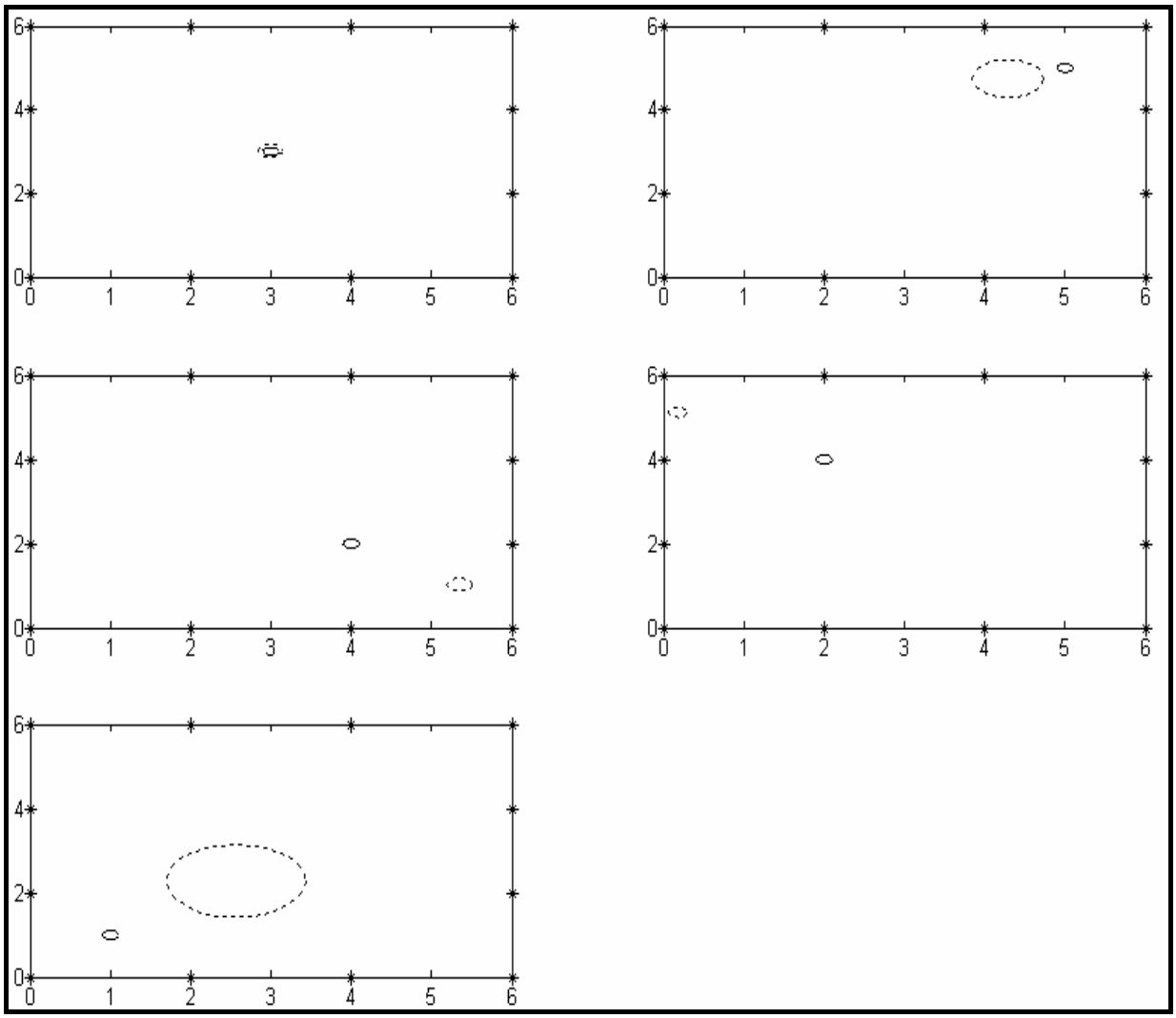

Figura 23 – Resultados obtidos pela ANN para 25 sensores

Tabela 1 – Resultados da ANN com 25 sensores

Real			**Encontrado**		
xc	***yc***	***r***	***xc***	***yc***	***r***
1	1	0,10	2,5655	2,2895	0,8644
2	4	0,10	0,1752	5,1169	0,1204
3	3	0,10	2,9894	3,0182	0,1540
4	2	0,10	5,3467	1,0279	0,1522
5	5	0,10	4,2912	4,7397	0,4437

Na Figura 24 e na Tabela 2 estão os resultados obtidos quando o número de sensores na placa é diminuído para 15 sensores. Pode ser verificado que há dificuldade em encontrar o raio correto do furo em todos os exemplos como no caso de 25 sensores, além da localização exata do furo não ter garantia de ser encontrada.

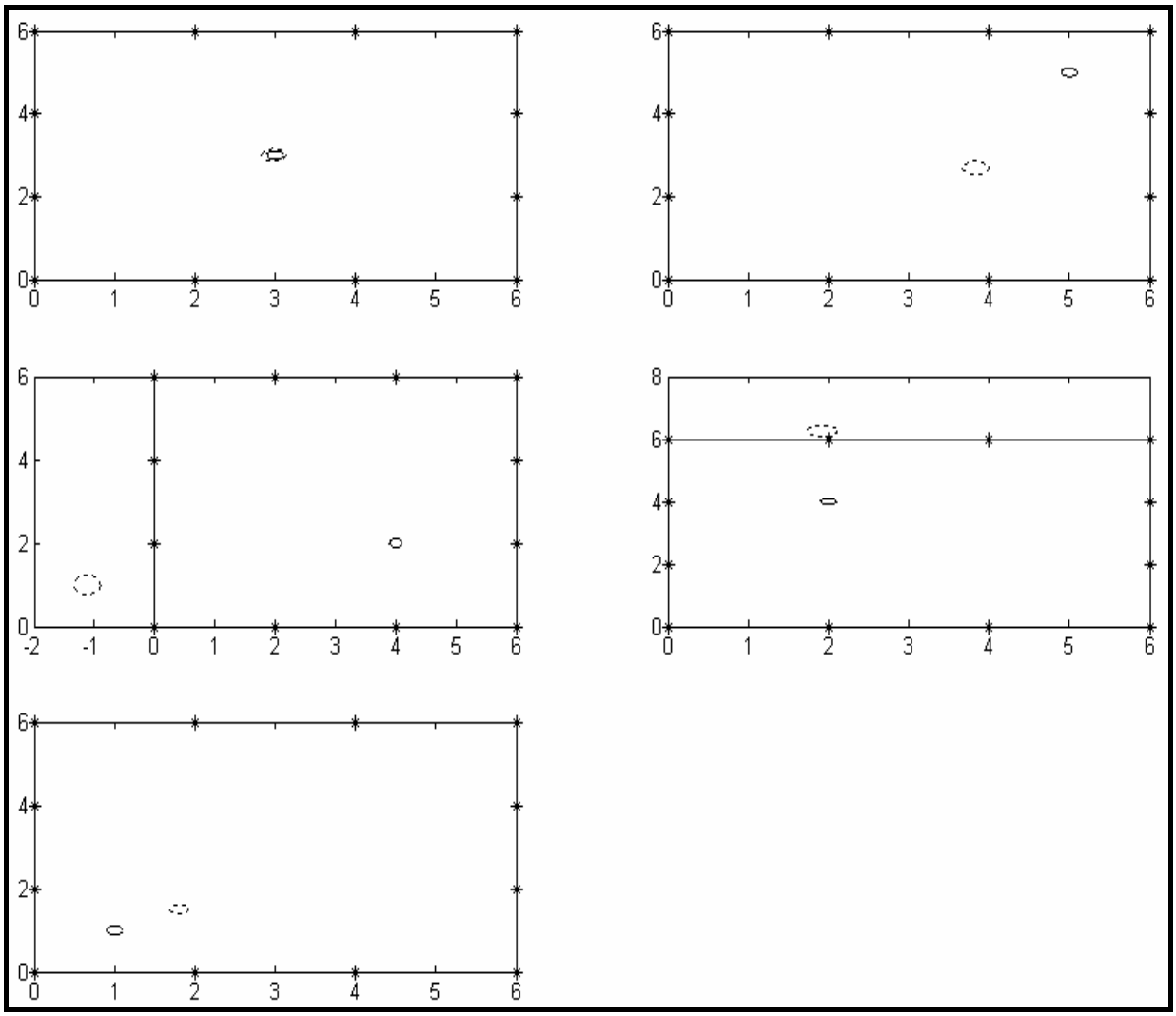

Figura 24 – Resultados obtidos pela ANN para 15 sensores

Tabela 2 – Resultados da ANN com 15 sensores

Real			**Encontrado**		
xc	***yc***	***r***	***xc***	***yc***	***r***
1	1	0,10	1,8056	1,5115	0,1136
2	4	0,10	1,9207	6,2514	0,1850
3	3	0,10	2,9818	2,9983	0,1498
4	2	0,10	-1,1220	0,9980	0,2257
5	5	0,10	3,8281	2,6935	0,1660

Pelas Figuras 25 e 26, são visualizados os resultados encontrados para 9 e 5 sensores na placa, respectivamente. Nas Tabelas 3 e 4 estão apresentados os resultados encontrados pela ANN para a configuração de parâmetros dada.

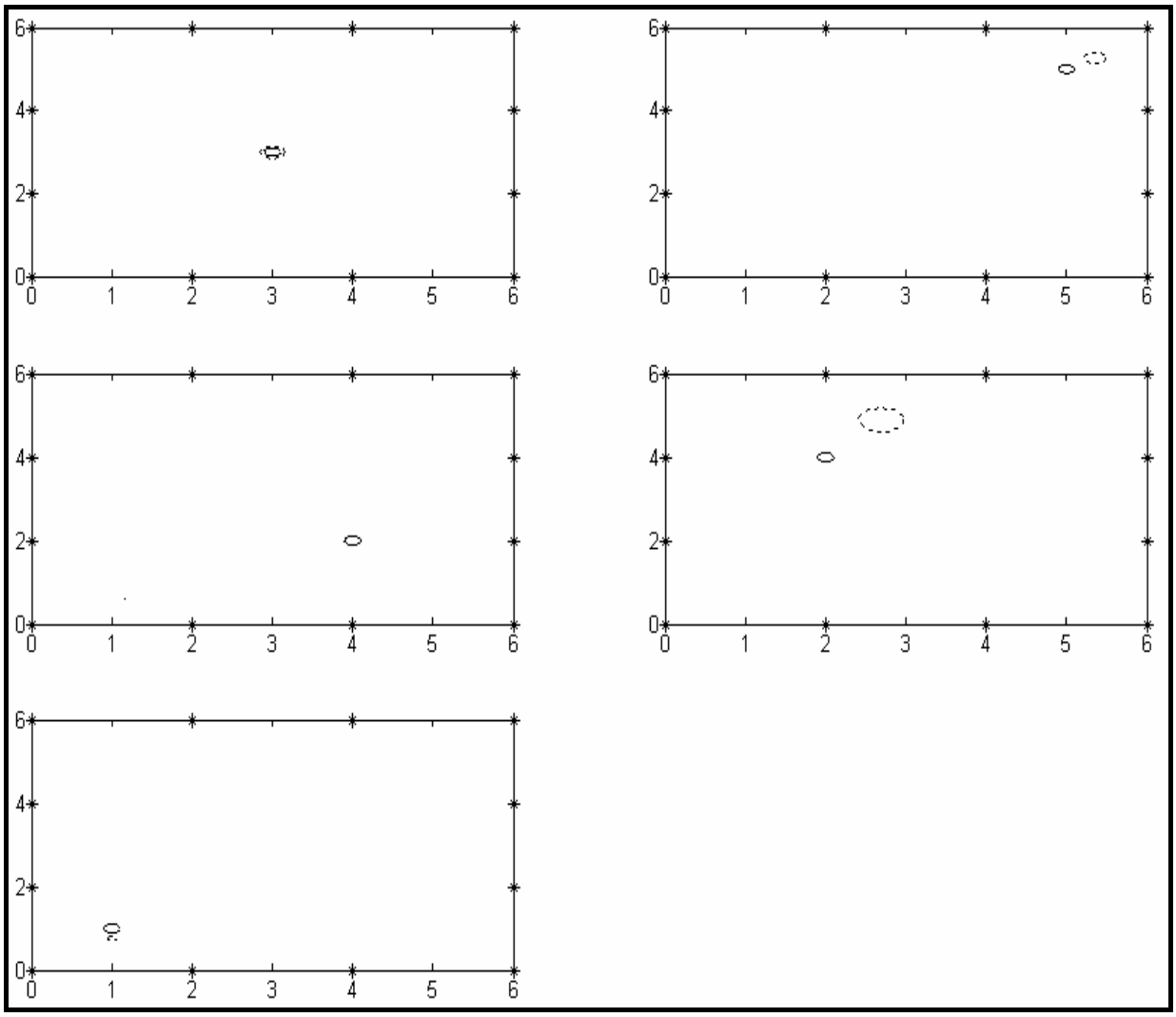

Figura 25 – Resultados obtidos pela ANN para 9 sensores

Tabela 3 – Resultados da ANN com 9 sensores

Real			**Encontrado**		
xc	***yc***	***r***	***xc***	***yc***	***r***
1	1	0,10	0,9946	0,7639	0,0640
2	4	0,10	2,6919	4,8912	0,2797
3	3	0,10	3,0003	2,9950	0,1513
4	2	0,10	1,1596	0,5967	0,0049
5	5	0,10	5,3517	5,2681	0,1377

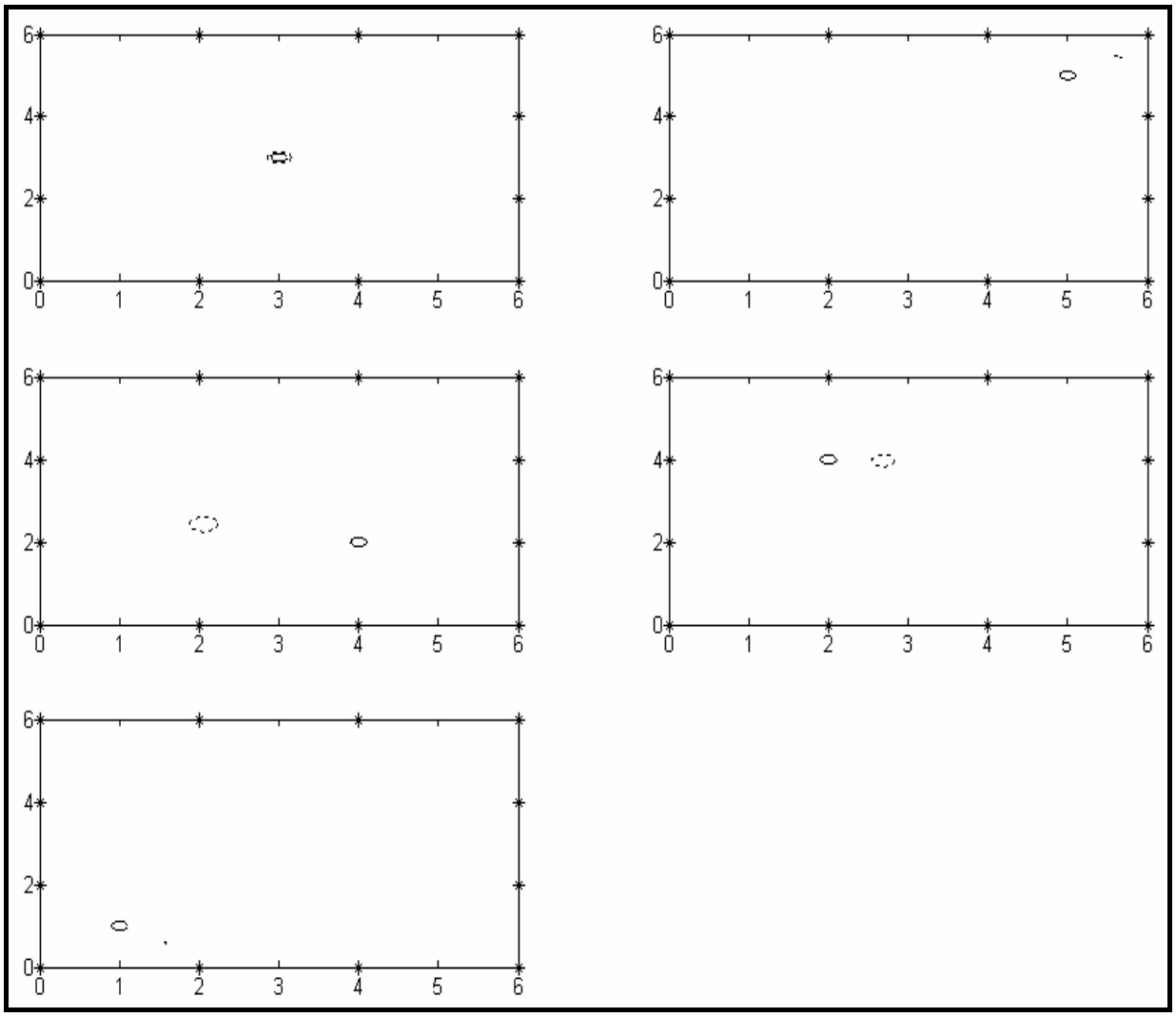

Figura 26 – Resultados obtidos pela ANN para 5 sensores

Tabela 4 – Resultados da ANN com 5 sensores

Real			**Encontrado**		
xc	***yc***	***r***	***xc***	***yc***	***r***
1	1	0,10	1,5880	0,5719	-0,0261
2	4	0,10	2,6888	3,9675	0,1415
3	3	0,10	3,0031	2,9986	0,1490
4	2	0,10	2,0524	2,4222	0,1781
5	5	0,10	5,6376	5,4556	0,0401

Da mesma forma que para o caso de 25 e 15 sensores na placa, os resultados não são satisfatórios e, uma nova configuração dos parâmetros e um novo conjunto de dados de entrada da ANN são necessários. Assim sendo, alterando os dados de entrada e alguns dos parâmetros de configuração da rede, novos resultados são obtidos. Assim como na

configuração dos parâmetros do GA, na ANN esta tarefa é árdua e necessita de constantes análises dos resultados obtidos até que um resultado satisfatório seja alcançado. Então, além de fornecer à rede os valores da diferença de potencial normalizada nos pontos representando os sensores para os furos com um raio de 0,15 *cm*, os valores da diferença de potencial normalizada para os furos com um raio de 0,05 *cm* são dados. Os resultados obtidos pela ANN após seu treinamento podem ser visualizados na Figura 27, considerando 25 sensores e estão sumarizados na Tabela 5. Os parâmetros de configuração alterados foram os seguintes:

- ✓ Número de neurônios na camada intermediária: 4
- ✓ Número de iterações: 5.000
- ✓ Taxa de aprendizado: 0.05

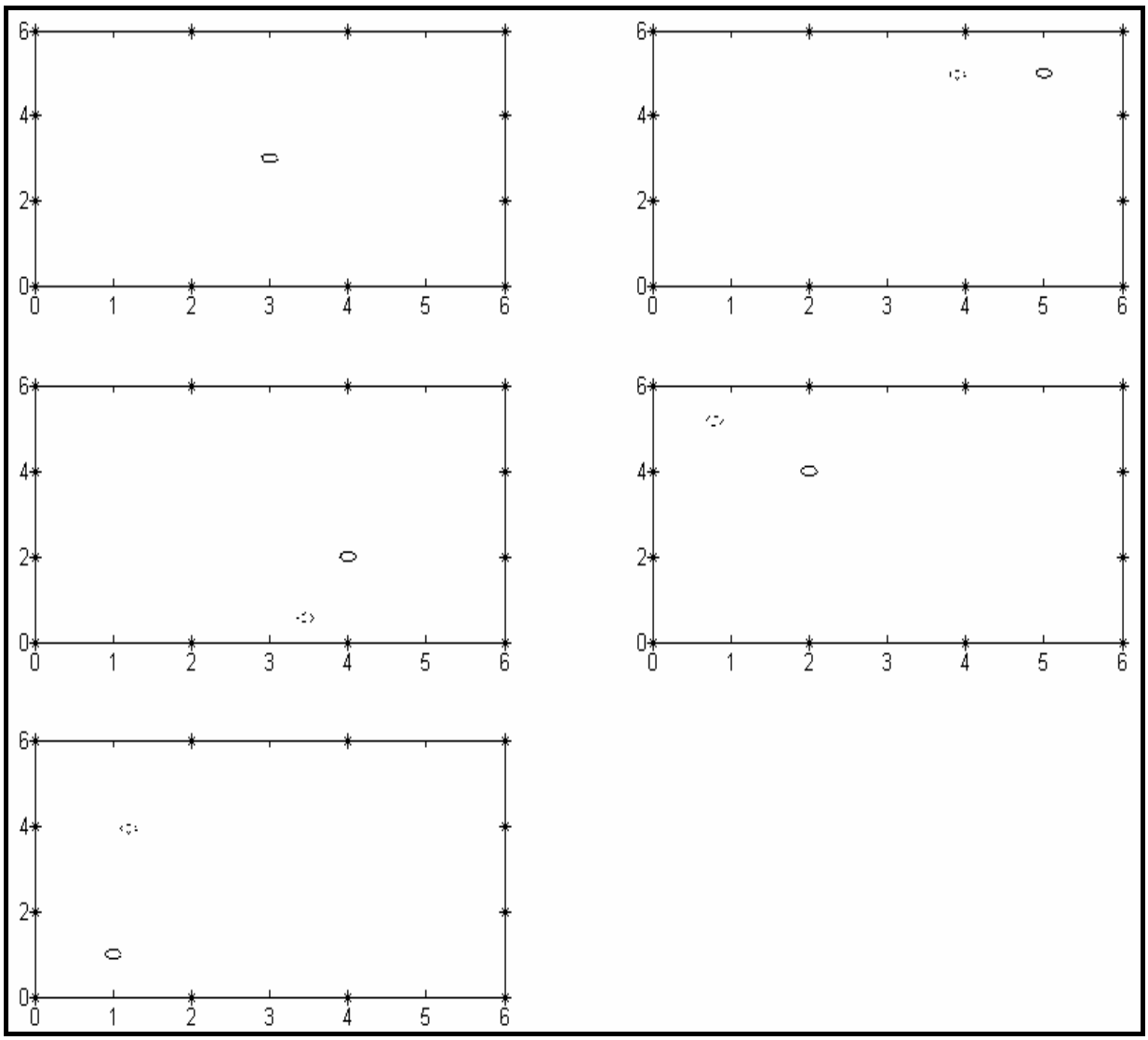

Figura 27 – Novos resultados obtidos pela ANN para 25 sensores

Tabela 5 – Resultados da ANN com 25 sensores e novos dados de entrada

Real			**Encontrado**		
xc	***yc***	***r***	***xc***	***yc***	***r***
1	1	0,10	1,2000	3,9294	0,0992
2	4	0,10	0,7924	5,1801	0,1004
3	3	0,10	3,0035	3,0003	0,0992
4	2	0,10	3,4568	0,5676	0,0994
5	5	0,10	3,8992	4,9665	0,0998

Como eram esperados, os resultados encontrados pela rede usando 25 sensores na placa para as novas informações dadas à rede apresentaram-se melhores do que para os dados anteriores. O programa de detecção de danos encontrou o valor do raio, mas ainda não encontrou a localização exata do dano, apenas na região central os resultados foram mais realistas. Na Figura 28 e na Tabela 6 estão os resultados encontrados pelo programa para 15 sensores. Novamente, o programa encontrou o valor do raio, porém apenas uma região próxima ao real foi encontrada.

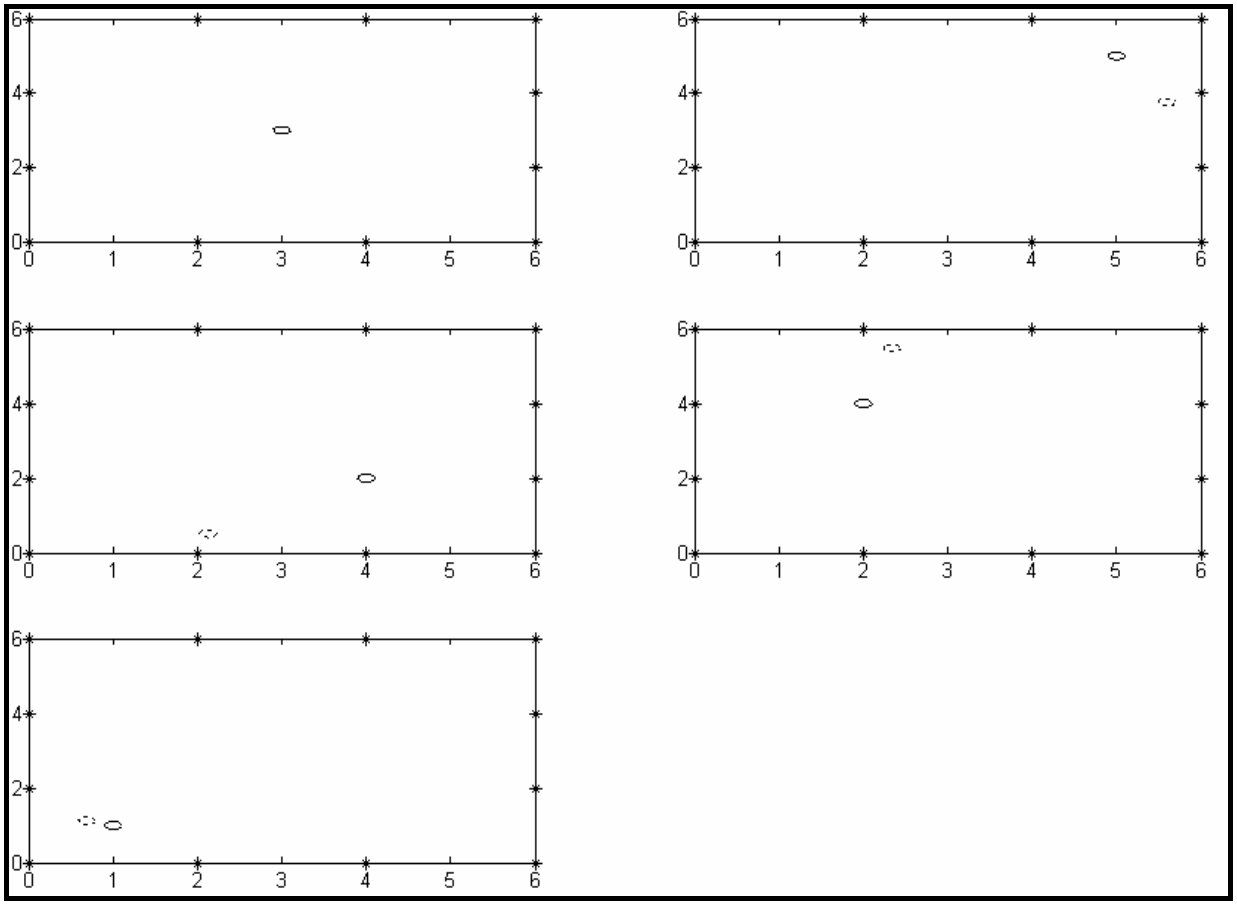

Figura 28 – Novos resultados obtidos pela ANN para 15 sensores

Tabela 6 – Resultados da ANN com 15 sensores e novos dados de entrada

Real			**Encontrado**		
xc	***yc***	***r***	***xc***	***yc***	***r***
1	1	0,10	0,6840	1,1139	0,1000
2	4	0,10	2,3356	5,4838	0,1000
3	3	0,10	2,9949	2,9977	0,1009
4	2	0,10	2,1225	0,5135	0,0994
5	5	0,10	5,5973	3,7565	0,1001

Com a diminuição do número de sensores na placa, como pode ser verificado pelas Figuras 29 e 30 e pelas Tabelas 7 e 8, a localização do dano não melhora pois há uma redução no domínio do problema. Além disso, o valor do raio começa a sofrer influência dessa redução de dados já que depende da distribuição dos sensores na placa tanto quanto da qualidade dos dados de entrada.

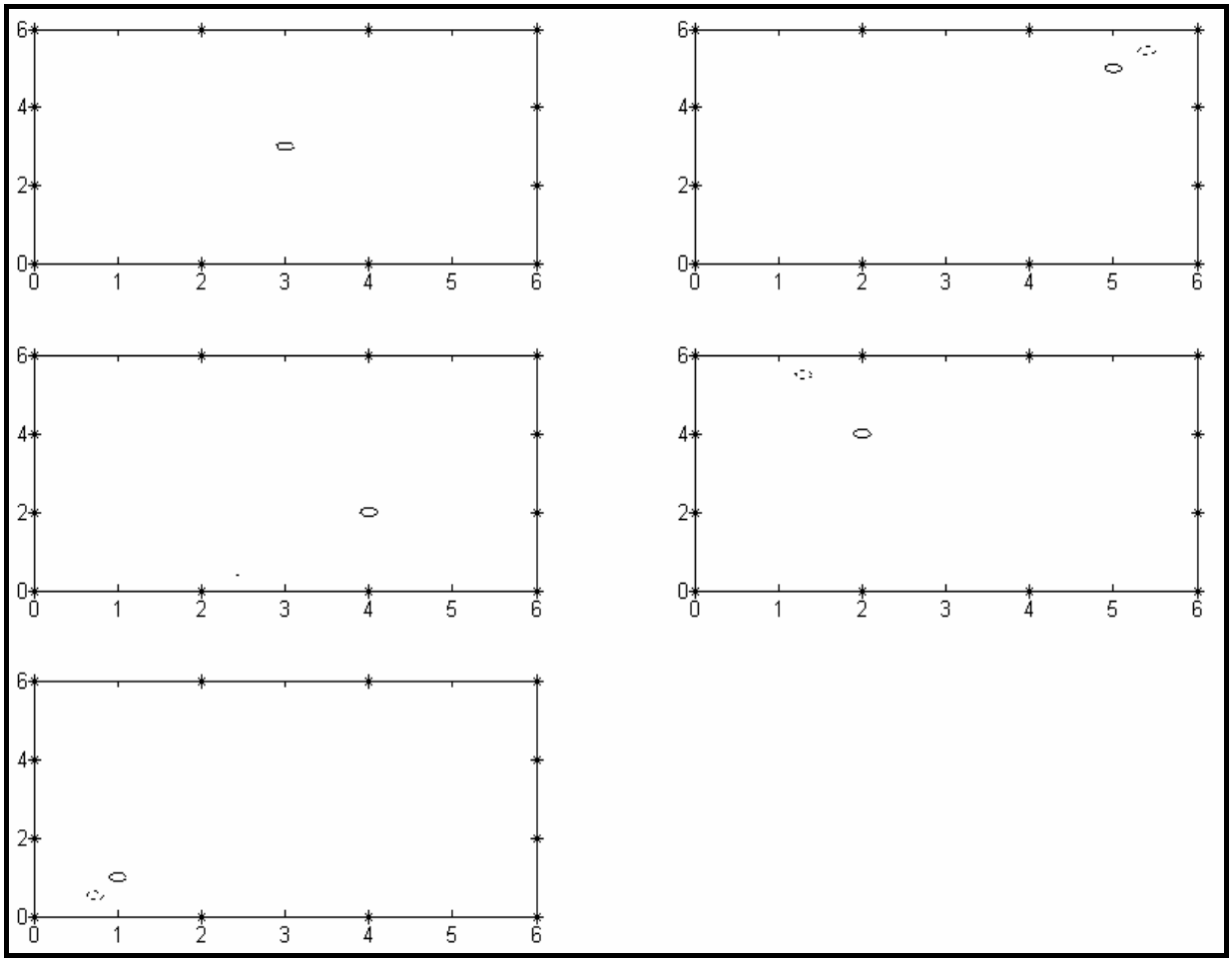

Figura 29 – Novos resultados obtidos pela ANN para 9 sensores

Tabela 7 – Resultados da ANN com 9 sensores e novos dados de entrada

Real			**Encontrado**		
xc	***yc***	***r***	***xc***	***yc***	***r***
1	1	0,10	0,7328	0,5264	0,1002
2	4	0,10	1,3012	5,4900	0,0976
3	3	0,10	2,9998	2,9973	0,1002
4	2	0,10	2,4224	0,4355	0,0224
5	5	0,10	5,3995	5,4423	0,0997

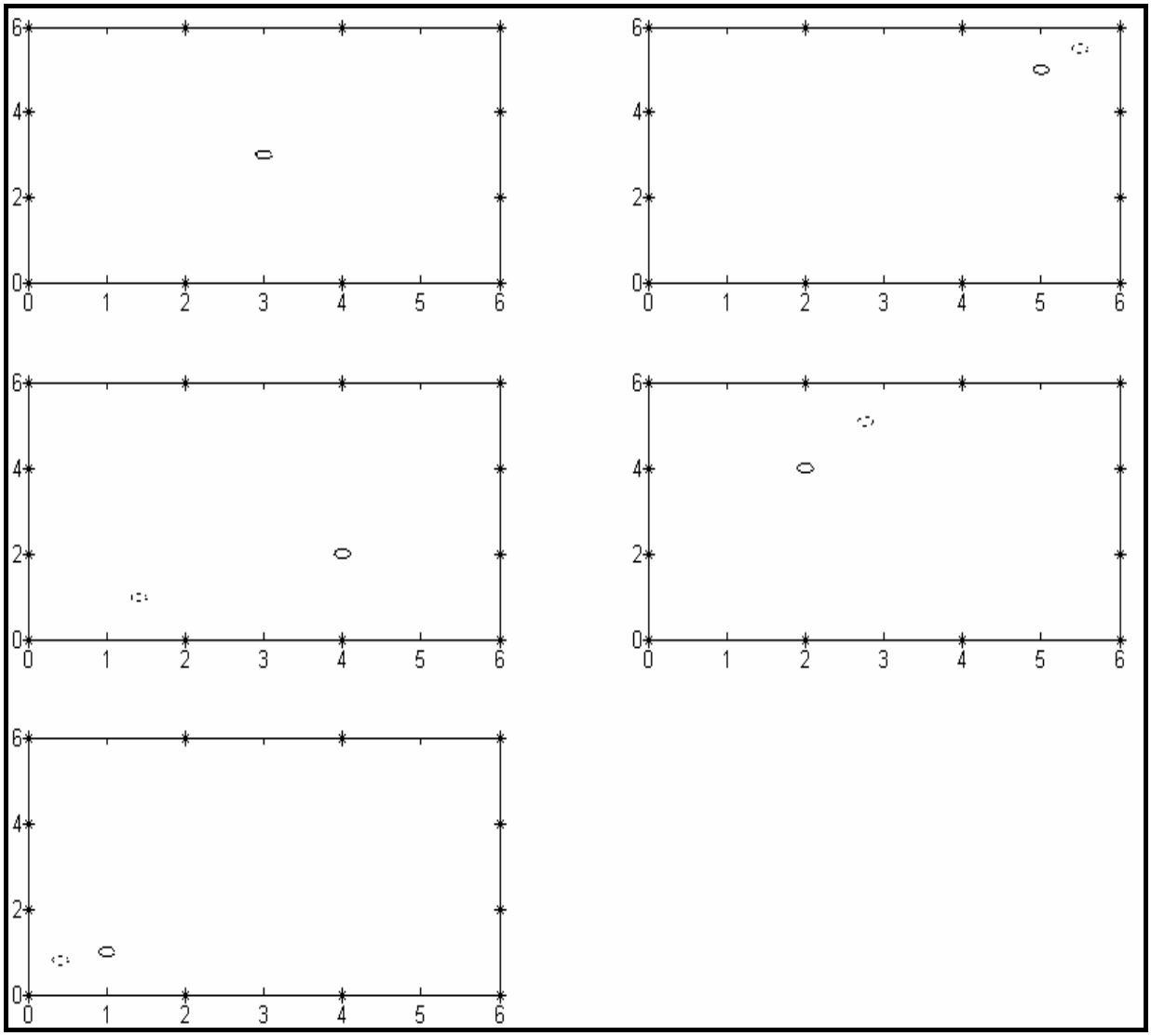

Figura 30 – Novos resultados obtidos pela ANN para 5 sensores

Tabela 8 – Resultados da ANN com 5 sensores e novos dados de entrada

Real			**Encontrado**		
xc	***yc***	***r***	***xc***	***yc***	***r***
1	1	0,10	0,4135	0,8019	0,1000
2	4	0,10	2,7664	5,0944	0,1000
3	3	0,10	3,0010	2,9959	0,1000
4	2	0,10	1,4138	0,9774	0,1000
5	5	0,10	5,4970	5,4926	0,1001

Nas Tabelas 9 e 10 estão apresentados a precisão dos resultados da ANN e o tempo de execução do programa de detecção de danos. Embora a precisão para o segundo caso (novos dados fornecidos à rede) fosse um pouco menor do que o primeiro caso (considerando apenas furos com raio 0,15 *cm*), o tempo gasto para o programa realizar os cálculos é aproximadamente a metade do tempo gasto no primeiro caso. Além disso, os resultados encontrados no segundo caso são melhores devido a melhora na qualidade dos dados fornecidos à ANN. Ainda observando as Tabelas 9 e 10, conforme o número de sensores na placa é reduzido, o tempo de execução do programa também diminui com era esperado.

Tabela 9 – Precisão e tempo de execução para o primeiro caso

Número de sensores	**Parada**	**Tempo de execução [s]**
25	$1{,}2885.10^{-9}$	84,1570
15	$3{,}7457.10^{-8}$	81,6410
9	$2{,}9383.10^{-7}$	80,1720
5	$6{,}8850.10^{-8}$	79,7340

Tabela 10 – Precisão e tempo de execução para o segundo caso

Número de sensores	**Parada**	**Tempo de execução [s]**
25	$6{,}3895.10^{-4}$	41,6560
15	$6{,}4596.10^{-4}$	41,3290
9	$6{,}3456.10^{-4}$	39,9380
5	$6{,}3128.10^{-4}$	39,0620

Nos exemplos simulados a seguir, os dados de entrada no programa de detecção de danos necessitaram ser modificados, permitindo que a ANN detectasse mais de um furo e

continuasse a detectar a presença de um furo apenas. O estudo foi focado na detecção de mais de um furo e como realizá-la, deste modo, foram considerados 25 sensores na placa e uma redução no número de sensores não foi feita. A Figura 31 (a) mostra a saída do programa de detecção de danos para um furo na posição (3;3) *cm* e raio de 0,10 *cm*. Na Figura 31 (b), está apresentado outro furo na posição (4;2) *cm* e de mesmo raio que o anterior. Na Figura 31 (c), estão apresentados dois furos numa placa nas posições (1;3) *cm* e (5;3) *cm*, ambos com um raio de 0,10 *cm*. Outro exemplo simulado foram dois furos numa placa, um na posição (1;1) *cm* e raio 0,15 *cm* e outro na posição (3;3) *cm* e raio 0,10 *cm*, conforme a Figura 31 (d). Finalmente, na Figura 31 (e) estão dois furos numa placa, um na posição (5;5) *cm* e raio 0,15 *cm* e outro na posição (3;3) *cm* e raio 0,10 *cm*.

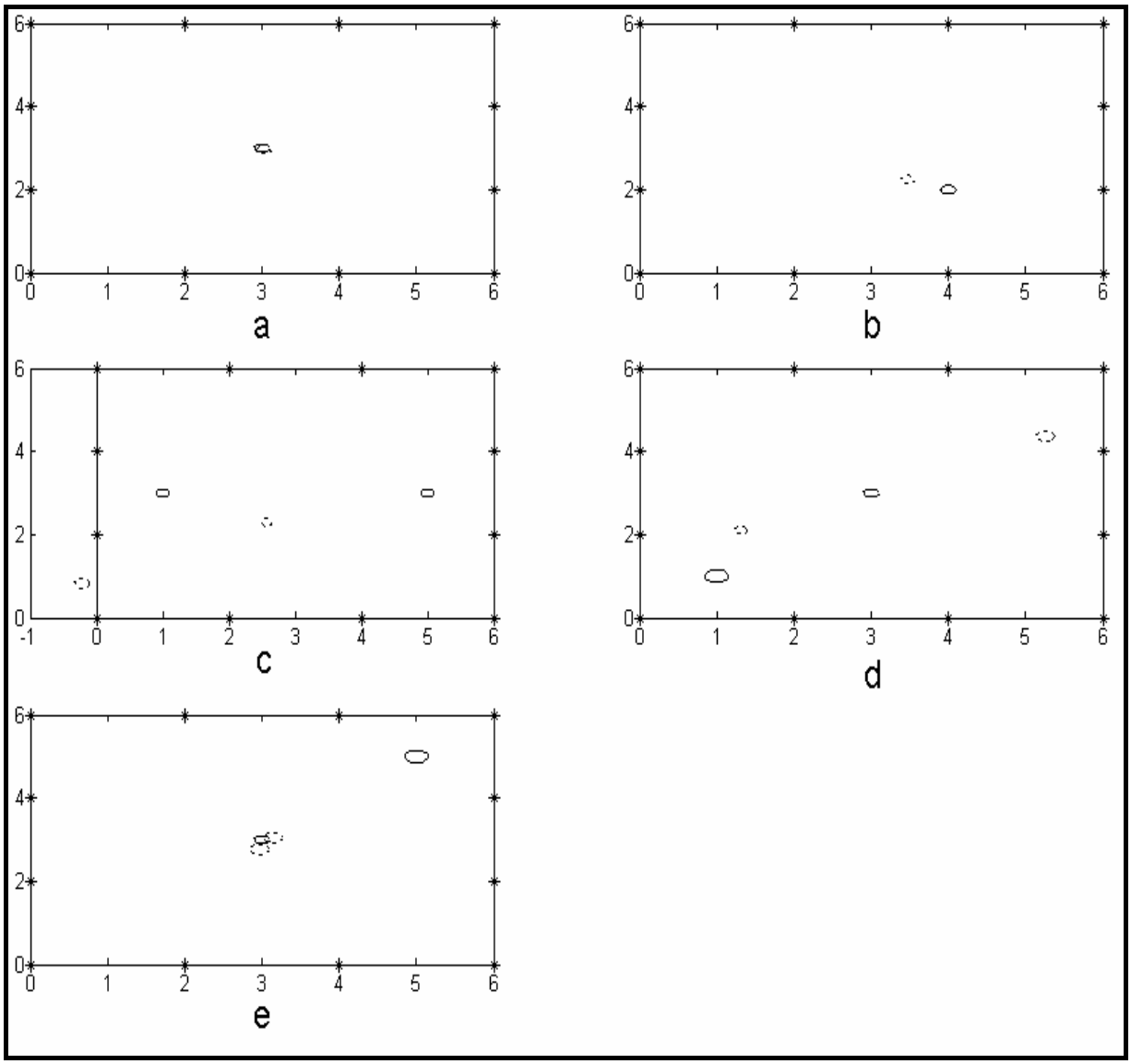

Figura 31 – Resultados do programa que detecta até dois furos na placa

A Tabela 11 apresenta os resultados obtidos pelo programa de detecção de danos que permite encontrar até dois furos na placa.

Tabela 11 – Resultados do programa que detecta até dois furos na placa

Real							**Encontrado**						
n	xc_1	yc_1	r_1	xc_2	yc_2	r_2	***n***	xc_1	yc_1	r_1	xc_2	yc_2	r_2
1	3	3	0,10	0	0	0	1	3,0326	2,9700	0,0957	-	-	-
1	4	2	0,10	0	0	0	1	3,4764	2,2467	0,0909	-	-	-
2	1	3	0,10	5	3	0,10	2	-0,2211	0,8259	0,1152	2,5722	2,3025	0,0915
2	1	1	0,15	3	3	0,10	2	1,3089	2,1078	0,0866	5,2559	4,3623	0,1179
2	5	5	0,15	3	3	0,10	2	2,9714	2,7625	0,1209	3,1428	3,0479	0,1216

A detecção de apenas um furo na placa já era uma tarefa difícil, a detecção de mais um furo apresentou dificuldades maiores. Como já foi mencionada, grande parte desta dificuldade é devida à má qualidade dos dados de entrada do programa de detecção de danos. Por fim, pelos exemplos apresentados nesta seção, os resultados dependem da qualidade dos dados de entrada da ANN e da escolha adequada dos parâmetros de configuração da mesma. Para prosseguir com a detecção de mais de um furo na placa por meio da ANN, o problema direto (os dados obtidos pelo BEM) deve ser melhorado. Novos carregamentos na placa e um novo BEM devem ser considerados, permitindo identificar não apenas furos circulares, mas também, elípticos e trincas nas estruturas.

Capítulo 6

CONCLUSÕES E PERSPECTIVAS FUTURAS

6.1 CONCLUSÕES

Neste trabalho duas técnicas de detecção de danos foram utilizadas, uma por meio do método de otimização global baseado em heurísticas, o algoritmo genético (GA) e, outra técnica por meio da identificação de parâmetros, as redes neurais artificiais (ANN's). O problema estudado neste trabalho foi um problema de transferência de calor, por isso o uso do método de elementos de contorno (BEM) para o potencial. O BEM para o potencial fornece as informações necessárias (valores do potencial em determinados pontos internos da placa) ao programa de detecção de danos. O enfoque do trabalho estava no estudo das técnicas de detecção de danos, GA e ANN e, além disso, o problema de transferência de calor foi utilizado em outra pesquisa desenvolvida anteriormente.

No problema de detecção de danos utilizando o GA foi verificado que este algoritmo é uma técnica trabalhosa, pois necessita analisar diversas vezes o problema para escolher os melhores parâmetros para configurá-lo. Além disso, o GA também apresenta um alto custo computacional devido às inúmeras avaliações da função *fitness* para que o algoritmo convirja. O programa de detecção de danos por meio do GA encontra a região de ocorrência do furo, porém apresenta dificuldade em encontrar o valor do raio do furo.

Uma das vantagens das técnicas de identificação de parâmetros sobre as técnicas de otimização global é que as primeiras possibilitam resolver o problema de detecção de danos mais rapidamente. Em comparação ao GA, o problema de detecção de danos resolvido por meio da ANN apresentou melhores resultados quanto a determinação do tamanho do raio, porém apresentou dificuldades em encontrar a região exata da ocorrência do dano. Possivelmente este fato foi devido a problemas de treinamento da rede, ou seja, à escolha dos parâmetros de configuração da rede e à escolha dos dados de entrada e saída da mesma. Inicialmente, nove posições diferentes de um furo de raio 0,15 *cm* foram consideradas na placa para formar o conjunto de dados de entrada da ANN e os resultados do programa de detecção de danos não foram satisfatórios. Obtendo novas informações pelo BEM para o potencial, um novo conjunto de dados foi formado, e uma nova configuração dos parâmetros da ANN foi necessária para encontrar resultados melhores para os dados de teste. Como foi verificado, o programa de detecção de danos para este novo conjunto de dados encontrou o valor do raio do furo, mas teve dificuldade em encontrar a posição exata deste furo.

Enfim, o GA é útil em localizar a região de ocorrência do furo enquanto que a ANN é útil para encontrar o tamanho deste furo. Considerando apenas as vantagens de cada técnica, uma abordagem híbrida poderia ser considerada. Neste tipo de abordagem, o GA poderia ser utilizado para encontrar a região de ocorrência do dano para então a ANN poder encontrar o tamanho exato deste dano. Mas um cuidado deve ser tomado quanto ao tamanho do furo, pois furos muito pequenos são difíceis de serem observados pelo programa de detecção de danos, principalmente quando estes furos estão próximos às bordas da placa.

6.2 PERSPECTIVAS FUTURAS

- Estudo sobre o posicionamento dos sensores na placa, visando reduzir o número de sensores na mesma;
- Substituição da variável potencial no BEM pela derivada da variável (componentes do gradiente), pois a derivada é mais sensível às mudanças na informação fornecida ao problema;
- Uso de BEM para o problema de elasticidade onde a variável é o deslocamento e sua derivada as deformações;

- Uso de uma abordagem híbrida, considerando as vantagens de cada uma das técnicas, ou seja, utilização do GA para encontrar a região de ocorrência do furo e a ANN para encontrar o tamanho exato deste furo;
- Uso do filtro de Kalman (KF) no problema de detecção de danos;
- Comparação entre os resultados obtidos pelas três técnicas, GA, KF e ANN's;
- Modificações no programa de detecção de danos para permitir que identificações de furos elípticos e trincas sejam feitas. Para o caso da presença de trincas na estrutura, é necessário o uso de BEM para a mecânica da fratura;
- Uso de outros métodos de otimização global, tais como, busca tabu, colônia de formigas e evolução diferencial para a comparação entre os resultados encontrados.

REFERÊNCIAS BIBLIOGRÁFICAS

BIGUS, J. P. (1996), *Data Mining with Neural Networks: Solving Business Problems from Application Development to Decision Support*, McGraw-Hill, 221p.

BREBBIA, C. A., DOMINGUEZ, J. (1992), *Boundary Elements: An Introductory Course*, 2nd Edition, McGraw-Hill Book Company, 314p.

BURCZYNSKI, T., BELUCH, W. (2001), "The identification of crack using boundary elements and evolutionary algorithms", *Engineering Analysis with Boundary Elements*, v 25, pp 313-322.

CECCHINI, A. (2005), *Damage detection and identification in sandwich composites using neural networks*, Dissertação de Mestrado em Engenharia Mecânica, Universidade de Porto Rico, Campus Mayagüez, 152p.

CHONG, E. K. P., ZAK, S. H. (2001), *An Introduction to optimization*, 2nd Edition, John Wiley & Sons, Inc., 495p.

DOUKA, E., LOUTRIDIS, S., TROCHIDIS, A. (2003), "Crack identification in beams using wavelet analysis", *International Journal of Solids and Structures*, v 40, pp 3557-3569.

ENGELHARDT, M., STAVROULAKIS, G. E., ANTES, H. (2006), "Crack and flaw identification in elastodynamics using Kalman filter techniques", *Computational Mechanics*, v 37, pp 249-265.

FARIA, M. C. P.; SOUZA, M. L. O. (2004), *Aplicação do filtro de Kalman para a estimação de estados em um problema de rastreamento*. INPE – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP. Disponível em: <http://iris.sid.inpe.br:1916/col/sid.inpe.br/marciana/2004/09.20.14.41/doc/aplicação%20 do%20filtro%20de%20kalman.pdf >. Acesso em: 25 abr. 2005

FREEMAN, J. A., SKAPURA, D. M. (1991), *Neural networks: algorithms, applications, and programming techniques*, Addison-Wesley Publishing Company, Inc., 401p.

GOLDBERG, D. E. (1998), *Genetic Algorithms in Search, Optimization and Machine Learning*, Massachusetts: Addison-Wesley Co, 372p.

LAGES, W. F. (2004a), *Descrição de sinais aleatórios*, Departamento de Engenharia Elétrica, Escola de Engenharia, Universidade Federal do Rio Grande do Sul Disponível em: <http://www.eletro.ufrgs.br/~fetter/ele00071/signals.pdf>. Acesso em: 18 abr. 2005.

LAGES, W. F. (2004b), *Filtro de Kalman*, Departamento de Engenharia Elétrica, Escola de Engenharia, Universidade Federal do Rio Grande do Sul. Disponível em: <http://www.eletro.ufrgs.br/~fetter/ele00071/kalman.pdf>. Acesso em: 18 abr. 2005.

LIANG, Y. C., HWU, C. (2001), "On-line identification of holes/cracks in composite structures", *Institute of Physics Publishing*, *Smart Materials and Structures*, v 10, pp 599-609.

LIU, S.W., HUANG, J.H., SUNG, J.C., LEE, C.C. (2002), "Detection of cracks using neural networks and computational mechanics", *Computer Methods in Applied Mechanics and Engineering*, v 191, pp 2831–2845.

MITCHELL, M. (1999), *An Introduction to Genetic Algorithms*, 5th Edition, MIT Press, Cambridge, Massachusetts-London, England, 209p.

NASH, S. G., SOFER, A. (1996), *Linear and Nonlinear Programming*, McGraw-Hill, 692p.

RAO, H. S., GHORPADE, V. G., MUKHERJEE, A. (2006), "A genetic algorithm based back propagation network for simulation of stress–strain response of ceramic-matrix-composites", *Computers and Structures*, v 84, pp 330–339.

SILVA, C. C., PEREIRA, J. A. (2005), *Implementação e análise de localização em robótica móvel por filtro de Kalman*, Instituto Tecnológico de Aeronáutica (ITA), Divisão de Ciência da Computação, São José dos Campos, SP. Disponível em: <http://www.bibl.ita.cta.br/ixencita/artigos/COMP_Claudiney.pdf>. Acesso em: 18 abr. 2005.

SIMON, D. (2001), "Kalman Filtering", *Embedded Systems Programming*, pp 72-79.

SPALL, J. C. (2003), *Introduction to stochastic search and optimization: estimation, simulation, and control*, Cap. 9 (Evolutionary Computation I: Genetic Algorithms), John Wiley & Sons, Inc., 595p.

STRAVOULAKIS, G. E., ANTES, H. (1998), "Flaw identification in elastomechanics: BEM simulation with local and genetic optimization", *Structural Optimization*, *Springer-Verlag*, v 16, pp 162-175.

TRONCOSO, J. F. A. (2004), *Estimación de parámetros para su aplicación a detección y diagnóstico de fallas*, Dissertação de Mestrado em Engenharia, Departamento de Ingeniería Eléctrica, Escuela de Ingeniería, Pontificia Universidad Católica de Chile, Santiago, Chile, 189p.

WELCH, G., BISHOP, G. (2004), *An Introduction to the Kalman Filter*, Department of Computer Science, University of North Carolina at Chapel Hill, NC 27599-3175. Disponível em: <http://www.cs.unc.edu/~welch/media/pdf/kalman _intro.pdf>. Acesso em: 25 abr. 2005.

Anexo A

PROGRAMA DO MÉTODO DIRETO POR MEIO DO MÉTODO DE ELEMENTOS DE CONTORNO

- **Programa que gera os dados para um furo apenas:**

```
%*************************************************************************
%preparando a area de trabalho
clc
clear
close all
%
% ******************** VARIAVEIS ************************
%
% npi: numero de pontos internos
% xi: coordenadas dos pontos internos
% x: coordenadas dos nos
% kode: 0 -> potencial prescrito;
% 1 -> fuxo prescrito
% fi: condiçoes iniciais de contorno
% n_placa: numero de nos da placa
% n_furo: numero de nos do furo
% r: raio do furo
% ixc: abscissa do centro do circulo
% iyc: ordenada do centro do circulo
% b: base da placa
% h: altura da placa
%
%*************************************************************************

format long
n_furo = 24;

%RAIO DO FURO
% r = 0.15; %dos dados.txt
r = 0.05; %dos dados_2.txt
% r = 0.10; %medido
```

```
k=1;

% DETERMINACAO DA POSICAO DO FURO (ixc,jyc)
for ixc = 0.5:2.5:5.5
    for jyc = 0.5:2.5:5.5
         ixc; %abscissa do centro do circulo
         jyc; %ordenada do centro do circulo
         % ixc = 5; %abscissa do centro do circulo: 3, 5, 4, 2, 1
         % jyc = 5; %ordenada do centro do circulo: 3, 5, 2, 4, 1
         disp('_____________________________________________________________________
          _____________________________');

         disp('PLACA COM FURO');
         x = determ_furo(n_furo,r,ixc,jyc);

         figure(1)

         %COM FURO
         [title,n,npi,x,kode,fi,xi,n_placa] = inputpc_PlacaFuro_25(x);
         % [title,n,npi,x,kode,fi,xi,n_placa] = inputpc_PlacaFuro_15(x);
         % [title,n,npi,x,kode,fi,xi,n_placa] = inputpc_PlacaFuro_09(x);
         % [title,n,npi,x,kode,fi,xi,n_placa] = inputpc_PlacaFuro_05(x);
         fprintf('Coordenadas do centro do furo: (%0.1f;',ixc);
         fprintf('%0.1f)\n',jyc);
         [pc,fi,dfi,xi,fipi] =
         poconbe_v5(title,n,npi,x,kode,fi,xi,n_placa);

         hold on

         disp(' ')
         disp('_____________________________________________________________________
         _____________________________');

         disp('PLACA SEM FURO');
         %SEM FURO

         figure(2)

         [title,n,npi,x,kode,fi,xi,n_placa]=inputpc_SEMfuro_25;
         % [title,n,npi,x,kode,fi,xi,n_placa]=inputpc_SEMfuro_15;
         % [title,n,npi,x,kode,fi,xi,n_placa]=inputpc_SEMfuro_09;
         % [title,n,npi,x,kode,fi,xi,n_placa]=inputpc_SEMfuro_05;
         [pc2,fi2,dfi2,xi2,fipi2] = poconbe_v5(title,n,npi,x,kode,fi,xi,
         n_placa);
         % load placaSemFuro;

         fprintf('\n')
         disp('_____________________________________________________________________
         _____________________________');

         disp('DIFERENCA ENTRE OS POTENCIAIS DA PLACA SEM E COM FURO');
         diferenca1 = abs(fipi2 - fipi);

         % normalizando o resultado da diferenca entre os potenciais
         [maior_pri,indmaior1] = max(diferenca1);
         dif_norm(:,1) = diferenca1/maior_pri;

         fprintf('\n')
         fprintf('%s\n\n','INTERNAL POINTS ')
         disp(' POT_1 - placa sem furo;')
         disp(' POT_2 - placa com furo;')
         fprintf('\n')
```

```
          fprintf('%s\n',' X  Y  POT_1  POT_2  DIF.POT.  DIF.POT.NORM.')
          for i=1:npi
               fprintf('\n' )
               fprintf('%1.4f\t',xi(i,1))
               fprintf('%9.4f\t',xi(i,2))
               fprintf('%9.4f\t',fipi2(i))
               fprintf('%9.4f\t',fipi(i))
               fprintf('%9.4f\t',diferenca1(i))
               fprintf('%10.4f\t',dif_norm(i))
          end
          fprintf('\n')
          disp('_____________________________________________________________
          _____________________________');

          % vx3(:,k) = fipi; %potencial placa com furo
          % vx4(:,k) = diferenca1; %diferenca entre os potenciais
          vx1(:,k) = ixc; %abscissa do centro do circulo
          vx2(:,k) = jyc; %ordenada do centro do circulo
          vx5(:,k) = dif_norm; %diferenca entre os potenciais normalizada
          % vx6(:,k) = xi(:,1); %vetor com as abscissas dos pontos internos
          % vx7(:,k) = xi(:,2); %vetor com as ordenadas dos pontos internos
          vx8(:,k) = r; %vetor com o raio do furo
          vx9(:,k) = 1; %vetor com o numero de furos

          F(k) = getframe;
          k = k+1;

     end
end


%são 9 (COLUNAS) posições de furos diferentes e 25 (ou 15, ou 9, ou 5)
%(LINHAS) sensores
% Obs.: cada posição de furo e os parametros do programa sao gravados em
% linha
%-------------------------------------------------------------------------
%AQUI COLOCA A FORMATAÇÃO DOS DADOS GRAVADOS
%-------------------------------------------------------------------------
for ij = 1:(k-1)
     % %para o programa q detecta apenas 1 furo
     % y(:,ij) = [vx5(:,ij); vx9(:,ij); vx1(:,ij); vx2(:,ij); vx8(:,ij)]
     %-------------------------------------------------------------------
     ----
     %para o programa q detecta 1 ou 2 furos
     y(:,ij) = [vx5(:,ij); vx9(:,ij); vx1(:,ij); vx2(:,ij); vx8(:,ij); 0;
     0;0];
     %-------------------------------------------------------------------
     ----
end

%VARREDURA: r = 0.15
% fid = fopen('dados25.txt','wt');
% fid = fopen('dados15.txt','wt');
% fid = fopen('dados09.txt','wt');
% fid = fopen('dados05.txt','wt');
%VARREDURA: r = 0.05
% fid = fopen('dados25_2.txt','wt');
% fid = fopen('dados15_2.txt','wt');
% fid = fopen('dados09_2.txt','wt');
% fid = fopen('dados05_2.txt','wt');

%-------------------------------------------------------------------------
%PARA PROG. DETECATA 1 OU 2 FUROS
```

```
fid = fopen('dados25_2f_5.txt','wt');
%-------------------------------------------------------------------------
%para 25 ptos + 7 parametros
fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f
%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4
f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f %11.4f \n',y);
%-------------------------------------------------------------------------

%MEDIDOS
% fid = fopen('f33r010_25ptos.txt','wt');
% fid = fopen('f33r010_15ptos.txt','wt');
% fid = fopen('f33r010_09ptos.txt','wt');
% fid = fopen('f33r010_05ptos.txt','wt');
% fid = fopen('f24r010_25ptos.txt','wt');
% fid = fopen('f55r010_09ptos.txt','wt');
% fid = fopen('f55r010_05ptos.txt','wt');

%-------------------------------------------------------------------------
%para 25 ptos + 4 parametros
% fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f
%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f \n',
y);
%-------------------------------------------------------------------------
%para 15 ptos + 4 parametros
% fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f
\n',y);
%-------------------------------------------------------------------------
%para 09 ptos + 4 parametros
% fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f %11.4f %11.4f %11.4f %11.4f \n',y);
%-------------------------------------------------------------------------
%para 05 ptos + 4 parametros
% fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f \n',y);
%-------------------------------------------------------------------------
fclose(fid)
%-------------------------------------------------------------------------
```

- **Programa que gera os dados para dois furos:**

```
%*************************************************************************
%preparando a area de trabalho
clc
clear
close all
% ******************** VARIAVEIS ************************
%
% npi: numero de pontos internos
% xi: coordenadas dos pontos internos
% x: coordenadas dos nos
% kode: 0 -> potencial prescrito;
% 1 -> fuxo prescrito
% fi: condiçoes iniciais de contorno
% n_placa: numero de nos da placa
% n_furo: numero de nos do furo
% r: raio do furo
% ixc: abscissa do centro do circulo
```

```
% iyc: ordenada do centro do circulo
% b: base da placa
% h: altura da placa
%
%*************************************************************************

format long

r2 = 0.10;
r1 = 0.15;

k=1;

% DETERMINACAO DA POSICAO DO FURO (ixc,jyc)
for ixc1 = 0.5:2.5:5.5
    for jyc1 = 0.5:2.5:5.5
        ixc1; %abscissa do centro do circulo
        jyc1; %ordenada do centro do circulo
        ixc2 = 3; %abscissa do centro do circulo
        jyc2 = 3; %ordenada do centro do circulo
        % ixc1 = 5; %abscissa do centro do circulo
        % jyc1 = 5; %ordenada do centro do circulo: 2, 3
        % ixc2 = 3; %abscissa do centro do circulo: 4, 5, 3.5
        % jyc2 = 3; %ordenada do centro do circulo: 5, 3, 3.5
        disp('_____________________________________________________________________
        _____________________________');

        disp('PLACA COM FURO');

        x1 = determ_furo(n_furo,r1,ixc1,jyc1);
        x2 = determ_furo(n_furo,r2,ixc2,jyc2);

        figure(1)

        %COM FURO
        fprintf('Coordenadas do centro do furo 1: (%0.1f;' ,ixc1);
        fprintf('%0.1f)\n',jyc1);
        fprintf('Coordenadas do centro do furo 2: (%0.1f;' ,ixc2);
        fprintf('%0.1f)\n',jyc2);
        [title,n,npi,x,kode,fi,xi,n_placa]=inputpc_PlacaCom2Furo_v7(x1,x2)
        [pc1,fi1,dfi1,xi1,fipi1] = poconbe_v5(title,n,npi,x,kode,fi,xi,
        n_placa);

        hold on

        disp(' ')

        disp('_____________________________________________________________________
        _____________________________');

        disp('PLACA SEM FURO');
        %SEM FURO

        figure(2)

        [title,n,npi,x,kode,fi,xi,n_placa]=inputpc_SEMfuro_25;
        [pc2,fi2,dfi2,xi2,fipi2] = poconbe_v5(title,n,npi,x,kode,fi,xi,
        n_placa);
        fprintf('\n')
        disp('_____________________________________________________________________
        _____________________________');

        disp('DIFERENCA ENTRE OS POTENCIAIS DA PLACA SEM E COM FURO');
```

```
        diferenca1 = abs(fipi2 - fipi1);

        % normalizando o resultado da diferenca entre os potenciais
        [maior_pri,indmaior1] = max(diferenca1);
        dif_norm(:,1) = diferenca1/maior_pri;

        fprintf('\n')
        fprintf('%s\n\n','INTERNAL POINTS ')
        disp(' POT_1 - placa sem furo;')
        disp(' POT_2 - placa com furo;')
        fprintf('\n')
        fprintf('%s\n',' X  Y  POT_1  POT_2  DIF.POT.  DIF.POT.NORM.')
        for i=1:npi
            fprintf('\n' )
            fprintf('%1.4f\t',xi(i,1))
            fprintf('%9.4f\t',xi(i,2))
            fprintf('%9.4f\t',fipi2(i))
            fprintf('%9.4f\t',fipi1(i))
            fprintf('%9.4f\t',diferenca1(i))
            fprintf('%10.4f\t',dif_norm(i))
        end
        fprintf('\n')
        disp('_____________________________________________________________________
        _____________________________');

        % vx31(:,k) = fipi1; %potencial placa com furo
        % vx32(:,k) = fipi2; %potencial placa com furo
        vx4(:,k) = diferenca1; %diferenca entre os potenciais
        vx11(:,k) = ixc1; %abscissa do centro do circulo 1
        vx21(:,k) = jyc1; %ordenada do centro do circulo 1
        vx12(:,k) = ixc2; %abscissa do centro do circulo 2
        vx22(:,k) = jyc2; %ordenada do centro do circulo 2
        vx5(:,k) = dif_norm; %diferenca entre os potenciais normalizada
        % vx6(:,k) = xi(:,1); %vetor com as abscissas dos pontos internos
        % vx7(:,k) = xi(:,2); %vetor com as ordenadas dos pontos internos
        vx8(:,k) = r1; %vetor com o raio do furo
        vx9(:,k) = r2; %vetor com o raio do furo
        vx10(:,k) = 2; %vetor com o numero de furos

        F(k) = getframe;
        k = k+1;

    end
end

%-------------------------------------------------------------------------
%AQUI COLOCA A FORMATAÇÃO DOS DADOS GRAVADOS
%-------------------------------------------------------------------------
for ij = 1:(k-1)
        y(:,ij) = [vx5(:,ij); vx10(:,ij); vx11(:,ij); vx21(:,ij);
        vx8(:,ij);vx12(:,ij); vx22(:,ij); vx9(:,ij)];
end

%VARREDURA:
fid = fopen('dados25_2f.txt','wt' );
% fid = fopen('dados25_2f_2.txt','wt');
% fid = fopen('dados25_2f_3.txt','wt');
% fid = fopen('dados25_2f_32.txt','wt');
%MEDIDOS
% fid = fopen('f13r010_53r010_25ptos.txt','wt');
% fid = fopen('f11r010_33r010_25ptos.txt','wt');
% fid = fopen('f11r015_33r010_25ptos.txt','wt');
% fid = fopen('f55r015_33r010_25ptos.txt','wt');
```

```
%-------------------------------------------------------------------------
%para 25 ptos + 7 parametros
fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f
%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4
f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f %11.4f \n',y);
%-------------------------------------------------------------------------
%para 15 ptos + 7 parametros
% fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f
%11.4f %11.4f %11.4f \n',y);
%-------------------------------------------------------------------------
%para 09 ptos + 7 parametros
% fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f \n',y);
%-------------------------------------------------------------------------
%para 05 ptos + 7 parametros
% fprintf(fid,'%11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %11.4f %
11.4f %11.4f %11.4f %11.4f \n',y);
%-------------------------------------------------------------------------
fclose(fid)
%-------------------------------------------------------------------------
```

- **Funções utilizadas no método de elementos de contorno:**

```
% ****************** SUBROTINA inputpc _SEMfuro_25 ***********************
%
% contem os dados para a soluçao do problema
%
% ******************** VARIAVEIS ***************************************
%
% title: titulo do problema
% n: numero de nós
% npi: numero de pontos internos
% xi: coordenadas dos pontos internos
% x: coordenadas dos nos
% kode: 0 -> potencial prescrito;
%       1 -> fluxo prescrito
% fi: condiçoes iniciais de contorno
%
%*************************************************************************

function [title,n,npi,x,kode,fi,xi,n_placa] = inputpc_SEMfuro_25
disp(' ');
disp(' ');
title = 'Placa com furo usando elementos constantes - POCONBE';

%coordenadas da placa
x_placa = [0 0
     2 0
     4 0
     6 0
     6 2
     6 4
     6 6
     4 6
     2 6
     0 6
     0 4
     0 2];
```

```
x = x_placa;
n = length(x);

n_placa = n; %o numero total de nos e o mesmo para o numero de nos da placa

%coordenadas dos pontos internos
%-------------------------------------------------------------------------
% AQUI SE MUDA A POSICAO DOS SENSORES
%-------------------------------------------------------------------------
k = 1;
for i = 1:5
     for j = 1:5
          xi(k,1) = i;
          xi(k,2) = j;
          k = k + 1;
     end
end

npi = length(xi(:,1)); %numero de pontos internos

%condicoes iniciais de contorno (um no por elemento)
kode = [1 1 1 0 0 0 1 1 1 0 0 0]; %naum hah furo
fi = [0 0 0 0 0 0 0 0 0 300 300 300];

%*************************************************************************

%******************** SUBROTINA inputpc_PlacaFuro_25***********************
%
% DADOS DE ENTRADA
%
%********************** VARIAVEIS ****************************************
%
% title: titulo do problema
% npi: numero de pontos internos
% xi: coordenadas dos pontos internos
% x_placa: coordenadas dos nos
% x: coordenadas do furo
% kode: 0 -> potencial prescrito;
%       1 -> fluxo prescrito
% fi: condiçoes iniciais de contorno
% n: numero de nos (elementos) totais (placa + furo)
% n_placa: numero de nos (elementos) da placa
%
%*************************************************************************

function [title,n,npi,x,kode,fi,xi,n_placa] = inputpc_PlacaFuro_25(x)
disp(' ');
disp(' ');
title = 'Placa com furo usando elementos constantes - POCONBE';
n_placa = 12;

%coordenadas da placa
x_placa = [0 0
     2 0
     4 0
     6 0
     6 2
     6 4
     6 6
     4 6
     2 6
     0 6
```

```
     0 4
     0 2
     0 0]; %para fechar
x = [x_placa
     x(:,1) x(:,2)];
n = length(x);

%coordenadas dos pontos internos
%-------------------------------------------------------------------------
% AQUI SE MUDA A POSICAO DOS SENSORES
%-------------------------------------------------------------------------
k = 1;
for i = 1:5
     for j = 1:5
          xi(k,1) = i;
          xi(k,2) = j;
          k = k + 1;
     end
end

npi = length(xi(:,1)); %numero de pontos internos

%condicoes iniciais de contorno
kode = [1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
1 1 1 1 1 1 1 1 1]; %furo adiabático: fluxo=0
fi = [0 0 0 0 0 0 0 0 0 300 300 300 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
0 0 0 0 0 0 0 0 0];

%*************************************************************************

%******************** SUBROTINA inputpc_PlacaCom2Furo_v7 ******************
%
% contem os dados para a soluçao do problema
%
% ********************** VARIAVEIS ***************************************
%
% title: titulo do problema
% n: numero de nós
% ng: numero pontos de Gauss
% npi: numero de pontos internos
% xi: coordenadas dos pontos internos
% x: coordenadas dos nos
% kode: 0 -> potencial prescrito;
% 1 -> fuxo prescrito
% n_placa: numero de nos da placa
% fi: condiçoes iniciais de contorno
% n: numero de nos
% b: base da placa
% h: altura da placa
%
%*************************************************************************

function [title,n,npi,x,kode,fi,xi,n_placa]=inputpc_PlacaCom2Furo_v7(x1,x2)
disp(' ');
disp(' ');
title='Placa com furo usando elementos constantes - POCONBE';
n_placa = 12;

x_placa=[0 0
     2 0
     4 0
     6 0
```

```
     6 2
     6 4
     6 6
     4 6
     2 6
     0 6
     0 4
     0 2
     0 0]; %para fechar
x=[x_placa
     x1(:,1) x1(:,2)
     x2(:,1) x2(:,2)];
n=length(x);

%coordenadas dos pontos internos
%-------------------------------------------------------------------------
% AQUI SE MUDA A POSICAO DOS SENSORES
%-------------------------------------------------------------------------
k = 1;
for i = 1:5
     for j = 1:5
          xi(k,1)=i;
          xi(k,2)=j;
          k1 = k + 1;
     end
end

npi=length(xi); %numero de pontos internos

%contorno interno e contorno dos dois furos
kode=[1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]; %furo adiabático
fi= [0 0 0 0 0 0 0 0 0 300 300 300 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];

%*************************************************************************

%************************** PROGRAMA POCONBE ******************************
%
% Problema de potencial usando elementos constantes
% adaptado de Brebbia & Dominguez, 2nd Ed. - Cap2.4
%
%************************** VARIAVEIS *************************************
%
% fi: potencial na fronteira
% dfi: fluxo na fronteira
% fipi: potencial no ponto interior
%
%*************************************************************************

function [pc,fi,dfi,xi,fipi] = poconbe_v5(title,n,npi,x,kode,fi,xi,n_placa)

x(n+1,:)=x(1,:); % "fechando o contorno"
clf

cplot_v2 % plota o contorno discretizado

% Calculo das matrizes g e h
[g h pc]=ghmatpc(n,x);

% aplicando as condições de contorno
[g dfi]=aplybc(g,h,kode,fi,n);
```

```
% solução do sistema
dfi=g\dfi';

% reordenar os vetores de potencial e fluxo para impressao
[fi dfi]=reorder(fi,dfi,kode,n);

% calculo do potencial nos pontos internos
fipi=interpc(n,npi,x,xi,dfi,fi);

% imprimindo os resultados
print_v2(title,n,npi,pc,fi,dfi,xi,fipi,n_placa);

%*********************************************************************

%************************ SUBROTINA CPLOT ******************************

figure(1)

for p = 1:(n_placa+1)
     x_placa(p,:) = x(p,:);
end

hold on
plot(x_placa(:,1),x_placa(:,2),'o-')

if n_placa ~= n
     pp = 1;
     for p = (n_placa + 2):(n-1)
          x_furo(pp,:) = x(p,:);
          pp = pp + 1;
     end
     plot(x_furo(:,1),x_furo(:,2),'-')
end

for i=1:n
     a=sprintf('%2g',i);
     text(x(i,1)+0.05,x(i,2)-0.05,a,'Color' ,'r');
end
hold on
plot(xi(:,1),xi(:,2),'+g')

%*********************************************************************

%************************ SUBROTINA GHAMATPC ***************************

function [g,h,pc]=ghmatpc(n,x)

epi=0; % epi=0 não é cálculo de ponto interno
for i=1:n, % varredura dos nós
     pc(i,:)=0.5*(x(i,:)+x(i+1,:)); % ponto de colocação
     for j=1:n, % varredura dos elementos
          if i == j
               %integ.analítica
               [haux gaux]=locinpc(i,j,pc(i,:),x(j,:),x(j+1,:));
          else
               %integ.numérica
               [haux gaux]=extinpc(i,j,pc(i,:),x(j,:),x(j+1,:),epi);
          end
          h(i,j)=haux;
          g(i,j)=gaux;
     end;
end
```

```
%*************************************************************************

%************************* SUBROTINA LOCINPC ****************************
%
% Calcula os elementos da matriz G, correspondendo as integrais ao longo
% dos elementos que incluem a singularidade
%
%*************************************************************************

function [h,g]=locinpc(i,j,pc,x1,x2)
L=norm(x2-x1); % comprimento do elemento
h=0.5;
g=L/(2*pi)*(1+log(2/L));

%*************************************************************************

%************************* SUBROTINA EXTINPC ****************************
%
% Retorna as integrais numericas do ponto de colocação pc até o elemento
% formado pelos pontos x1 e x2
%
%************************* VARIAVEIS ************************************
%
% ng: numero pontos de Gauss
% w: pesos dos pontos de Gauss
% e: coordenadas dos pontos de Gauss
% epi: flag para ponto interno
% pc: ponto de colocaçao
%
%*************************************************************************

function [h,g]=extinpc(i,j,pc,x1,x2,epi)

ng=4; % nr de pontos de integração
[w e]=gauss(ng);
L=norm(x2-x1); % comprimento do elemento
for z=1:ng,
     xc(z,:)=e(1,z).*(x1-x2)./2 +(x1+x2)./2; %mapeam.conforme
                                              [0 L] -> [-1 1]
     r(1,z)=norm(xc(z,:)-pc %dist. ponto de colocação ponto de integração
end

D=dist(pc,x1,x2); % distância do ponto de colocação ao elemento

if (i==j) & (epi~=1) %epi=1 flag para ponto interior
     h=0.5;
     g=L/(2*pi)*(1+log(2/L));
else
     Haux=-1./(2*pi*r.^2)*D.*w*L/2;
     Gaux=1./(2*pi).*log(1./r).*w*L/2;
     h=sum(Haux);
     g=sum(Gaux);
end

%*************************************************************************

%************************* SUBROTINA DIST *******************************
%
% Calcula a distAncia do ponto xp ate a reta formada por x1 e x2
%
%************************* VARIAVEIS ************************************
%
```

```
% xp: coordenadas do ponto de colocaçao
% x1: primeito nó do elemento a ser varrido
% x2: segundo nó do elemento a ser varrido
%
%*************************************************************************

function d=dist(xp,x1,x2)
xp=[xp 0];
x2=[x2 0];
x1=[x1 0];
a=(xp-x2);
b=(x1-x2)/norm(x1-x2);
d=norm(cross(a,b));
sig=sum(cross((x1-xp),(x2-xp)));
if sig < 0
     d=-d;
end

%*************************************************************************

%************************** SUBROTINA APPLYBC ****************************
%
% Reordena as colunas do sistema de acordo com as condicoes de contorno
% e forma a matriz coeficiente armazenada em G e o lado direito em DFI
%
%************************** VARIAVEIS ************************************
%
% h: matriz quadrada (n,n)
% g: matriz retangular (n,2*n)
%
%*************************************************************************

function [g,dfi]=aplybc(g,h,kode,fi,n);
for j=1:n,
     if kode(j) ~= 0
          for i=1:n,
               ch=g(i,j);
               g(i,j)=-h(i,j);
               h(i,j)=-ch;
          end
     end
end
dfi=h*fi';
dfi=dfi';

%*************************************************************************

%************************** SUBROTINA REORDER ****************************
%
% Armazena o potencial(fi) e o fluxo (dfi) nos respectivos vetores
%
%************************** VARIAVEIS ************************************
%
% fi: potencial do nó
% dfi: fluxo antes no nó
%
%*************************************************************************

function [fi,dfi]=reorder(fi,dfi,kode,n)
for i=1:n,
     if kode(i) ~= 0
          ch=fi(i);
          fi(i)=dfi(i);
```

```
          dfi(i)=ch;
     end
end

%*************************************************************************

%************************** SUBROTINA INTERPC ****************************
%
% Calcula os valores de potencial nos pontos internos
%
%************************** VARIAVEIS ************************************
%
% fipi: potencial no ponto interno
% xi: coordenadas dos pontos internos
%
%*************************************************************************

function fipi=interpc(n,npi,x,xi,dfi,fi)

clear g h;
epi=1; % epi=1 calculo de ponto interno
for i=1:npi,
     for j=1:n,
          [haux gaux]=extinpc(i,j,xi(i,:),x(j,:),x(j+1,:),epi); % integ.
          numérica
          h(i,j)=haux;
          g(i,j)=gaux;
     end;
end
% soluçao para pontos internos (Paris & Cannas pag 77, eq 3.4.18)
fipi=g*dfi-h*fi';

%*************************************************************************

%************************** SUBROTINA INTERPC ****************************
%
% Plota os resultados na tela
%
%*************************************************************************

function print_v2(title,n,npi,pc,fi,dfi,xi,fipi,n_placa)

% para o contorno
fprintf('%s\n\n',title)
fprintf('%s\n\n','RESULTS')
fprintf('%s\n\n','BOUNDARY NODES ')
fprintf('%s\n','NODE X Y POTENTIAL
POTENTIAL DERVATIVE')
for i=1:n
     if (i == (n_placa + 1))|(i == n)
          continue
     else
          fprintf('\n')
          fprintf('%.0f\t',i)
          fprintf('%15.4f\t',pc(i,1))
          fprintf('%15.4f\t',pc(i,2))
          fprintf('%15.4f\t',fi(i))
          fprintf('%15.4f\t',dfi(i))
     end
end

% para os pontos internos
fprintf('\n\n')
```

```
fprintf('%s\n\n','INTERNAL POINTS ')
fprintf('%s\n',' X Y POTENTIAL' )
for i=1:npi
     fprintf('\n' )
     fprintf('%15.4f\t',xi(i,1))
     fprintf('%15.4f\t',xi(i,2))
     fprintf('%15.4f\t',fipi(i))
end

%*************************************************************************
```

Anexo B

PROGRAMA DE DETECÇÃO DE DANOS UTILIZANDO ALGORITMOS GENÉTICOS

- **Programa de detecção de um furo usando o algoritmo genético:**

```
%************************************************************************
%
% PROGRAMA PARA ENCONTRA A POSICAO DO FURO E O SEU RAIO
%
%************************************************************************
clear
clc

%DADOS DE ENTRADA: diferenca de potenciais da placa com e sem furo
% vx4(:,k) - difpot (diferenca de potencial da placa sem e com furo)
% vx1(:,k) - xc (abscissa do centro do furo);
% vx2(:,k) - yc (ordenada do centro do furo).

load('Vx1234_r15_110med' ,'vx1','vx2','vx4'); %DIFERENCA POT. PLACA COM E
SEM FURO
vx11 = vx1;
vx21 = vx2;
vx51 = vx4;
max1 = max(vx51);

medido = [transpose(vx4(:,10))]; %DIFERENCA POT. PLACA COM FURO DE
PARAMETROS DESCONHECIDOS
n_medido = length(medido(1,:)); %comprimento do vetor solucao 'pop' do
melhor individuo encontrado

load('Vx1234_r03_110med' ,'vx1','vx2','vx4');
vx12 = vx1;
vx22 = vx2;
vx52 = vx4;
max2 = max(vx52);
```

```
load('Vx1234_r09_110med' ,'vx1','vx2','vx4'); %DIFERENCA POT. PLACA COM E
SEM FURO
vx14 = vx1;
vx24 = vx2;
vx54 = vx4;
max3 = max(vx54);

%NORMALIZANDO OS VALORES DA DIFERENCA DE POTENCIAIS INTERNOS NA PLACA.
maxTotal = max([max1 max2 max3]) %maximo valor da diferenca de potencial
vx51 = vx51/maxTotal;
vx52 = vx52/maxTotal;
vx54 = vx54/maxTotal;

%concatenando horizontalmente
vx13 = horzcat(vx11,vx12,vx14);
vx23 = horzcat(vx21,vx22,vx24);
vx53 = horzcat(vx51,vx52,vx54);

n_linhas = length(transpose(vx53(1,:))); %comprimento do vetor solucao
'pop' do melhor individuo encontrado
k=111;

tic; %starts a stopwatch timer.

%FORMANDO A POPULACAO DE INDIVIDUOS
for p = 1:(k-1)
     pop(p,1:(n_medido+3)) = [transpose(vx13(p)) transpose(vx23(p)) 0.15
     transpose(vx53(:,p))];
end
for p = k:(n_linhas-(k-1))
     pop(p,1:(n_medido+3)) = [transpose(vx13(p)) transpose(vx23(p)) 0.03
     transpose(vx53(:,p))];
end
for p = (n_linhas-(k-2)):n_linhas
     pop(p,1:(n_medido+3)) = [transpose(vx13(p)) transpose(vx23(p)) 0.09
     transpose(vx53(:,p))];
end

n_pop = length(pop(1,:)); %NUMERO DE VARIAVEIS

disp('_____________________________________________________________________
_____')
fprintf('\nENCONTRANDO O MÍNIMO DA FUNCAO USANDO O GA\n\n' )
for i = 1:5
     %CRIANDO A ESTRUTURA DE OPCOES DO GA E CALCULANDO O MINIMO
     options = gaoptimset('InitialPopulation',pop, 'PopulationSize',
     p,'Generations' ,75,'CrossoverFraction' ,0.8,'CrossoverFcn' ,
     {@crossoverheuristic,1.3},'PlotFcns',@gaplotbestf);
     [pop1, fval, reason, output] = ga(@funcional_J3_normaliz, n_pop,
     options)

     %parametros encontrados pelo GA
     x_furo(i) = pop1(1,1);
     y_furo(i) = pop1(1,2);
     raio(i) = pop1(1,3);
     fun_J(i) = fval;

     figure(2)
     plot_circulo(pop1(1,1),pop1(1,2),pop1(1,3))
     hold on
     %-------------------------------------------------------------------
     ----
```

```
    % AQUI SE MUDA O PLOT DO FURO REAL
    %-------------------------------------------------------------------------
    ----
    % plot_circulo2(3,3,0.06,2)
    plot_circulo2(4,5,0.06,2)
end

t1 = toc/i; %returns the elapsed time in t.
fprintf('\n\n');
fprintf('tempo de execução: %f segundos\n',t1);
fprintf('\n\n');

%calculando os valores médios e as incertezas com 99.7%
xmean = mean(x_furo);
xstd = std(x_furo);
xincert = 3*std(x_furo);
ymean = mean(y_furo);
ystd = std(y_furo);
yincert = 3*std(y_furo);
rmean = mean(raio);
rstd = std(raio);
rincert = 3*std(raio);
jmean = mean(fun_J);
jstd = std(fun_J);
jincert = 3*std(fun_J);

%saída na tela
fprintf('\n\nValores encontrados pelo GA (99.7%% confianca)\n')
fprintf('xc = %1.3f ',xmean)
fprintf('+- %1.3f\t',xincert)
fprintf('std %1.3f\t',xstd)
fprintf('\nyc = %1.3f ',ymean)
fprintf('+- %1.3f\t',yincert)
fprintf('std %1.3f\t',ystd)
fprintf('\nr = %1.3f ',rmean)
fprintf('+- %1.3f\t',rincert)
fprintf('std %1.3f\t',rstd)
fprintf('\nJ = %1.3f ',jmean)
fprintf('+- %1.3f\t',jincert)
fprintf('std %1.3f\t',jstd)
fprintf('\n\n')

function J = funcional_J3(pop)

% load('Vx1234_r06_p33','vx1','vx2','vx4'); %furo em (xc,yc) = (3,3)
load('Vx1234_r06_p45','vx1','vx2' ,'vx4' ); %furo em (xc,yc) = (4,5)
vx4 = vx4/250.4792; %maxTotal = max([max1 max2 max3]) = 250.4792
medido = [transpose(vx4)];

aux = length(pop(1,:));
n = length(pop(:,1));
for w = 1:n
    J(w) = 0.5*sum((medido - pop(w,4:aux)).^2);
end
%ENCONTRA A SOMA MINIMA, LOGO, ENCONTRA O DANO...
```

- **Programa de detecção de até dois furos usando o algoritmo genético:**

```
%*************************************************************************
%
% PROGRAMA PARA ENCONTRA A POSICAO DO FURO E O SEU RAIO
%
%*************************************************************************
clear all
clc
close all

%*************************************************************************
%DADOS DE ENTRADA: diferenca de potenciais da placa com e sem furo
%*************************************************************************
%placa com um furos
%*************************************************************************
load('Vx1234_r15_110med' ,'vx1','vx2','vx4');
vx11 = vx1; %coordenada x dos pontos internos
vx21 = vx2; %coordenada y dos pontos internos
vx51 = vx4; %diferenca de potencial
max1 = max(vx51);

medido = [transpose(vx4(:,10))];
n_medido = length(medido(1,:)); %comprimento do vetor solucao 'pop' do
melhor individuo encontrado
load('Vx_13r03_33_r15.mat','vx1' ,'vx2','vx4');
vx5med = vx4;
maxmed = max(vx5med); %usado na normalizacao de vetores

load('Vx1234_r03_110med' ,'vx1','vx2','vx4');
vx12 = vx1;
vx22 = vx2;
vx52 = vx4;
max2 = max(vx52);

load('Vx1234_r09_110med' ,'vx1','vx2','vx4');
vx14 = vx1;
vx24 = vx2;
vx54 = vx4;
max3 = max(vx54);

%*************************************************************************
%placa com dois furos
%*************************************************************************
load('VarreFuro1r15_furos2_33r03' ,'vx1' ,'vx2','vx4');
vx15 = vx1; %coordenada x dos pontos internos
vx25 = vx2; %coordenada y dos pontos internos
vx55 = vx4; %diferenca de potencial, respectivamente, furo 1 e furo 2
max4 = max(vx55);
load('VarreFuro1r03_furos2_33r15' ,'vx1' ,'vx2','vx4');
vx16 = vx1;
vx26 = vx2;
vx56 = vx4; %diferenca de potencial, respectivamente, furo 1 e furo 2
max5 = max(vx56);
load('VarreFuro1r03_furos2_33r03' ,'vx1' ,'vx2','vx4');
vx17 = vx1;
vx27 = vx2;
vx57 = vx4; %diferenca de potencial, respectivamente, furo 1 e furo 2
max6 = max(vx57);

%*************************************************************************
```

```
%NORMALIZANDO OS VALORES DA DIFERENCA DE POTENCIAIS INTERNOS NA PLACA.
%*************************************************************************
maxTotal = max([max1 max2 max3 max4 max5 max6 maxmed]) %maximo valor da
diferenca de potencial entre os valores calculados pelo BEM para r = 0.15,
0.09 e 0.03
vx51 = vx51/maxTotal;
vx52 = vx52/maxTotal;
vx54 = vx54/maxTotal;
vx55 = vx55/maxTotal;
vx56 = vx56/maxTotal;
vx57 = vx57/maxTotal;

%para 1 furo apenas
vx13 = horzcat(vx11,vx12,vx14);
vx23 = horzcat(vx21,vx22,vx24);
vx53 = horzcat(vx51,vx52,vx54);

%para 2 furos
vx1x1 = horzcat(vx15,vx16,vx17);
vx2y1 = horzcat(vx25,vx26,vx27);
vx523 = horzcat(vx55,vx56,vx57);

n_linhas1 = length(transpose(vx53(1,:))); %comprimento do vetor solucao
'pop' do melhor individuo encontrado
n_linhas2 = length(transpose(vx523(1,:))); %comprimento do vetor solucao
'pop' do melhor individuo encontrado

k=111;

tic; %starts a stopwatch timer.

%*************************************************************************
%FORMANDO A POPULACAO DE INDIVIDUOS (para 1 furo)
%*************************************************************************
for p = 1:(k-1)
    pop1(p,1:(n_medido+7)) = [1 transpose(vx13(p)) transpose(vx23(p)) 0 0
    0.15 0 transpose(vx53(:,p))];
end
for p = k:(n_linhas1-(k-1))
    pop1(p,1:(n_medido+7)) = [1 transpose(vx13(p)) transpose(vx23(p)) 0 0
    0.03 0 transpose(vx53(:,p))];
end
for p = (n_linhas1-(k-2)):n_linhas1
    pop1(p,1:(n_medido+7)) = [1 transpose(vx13(p)) transpose(vx23(p)) 0 0
    0.09 0 transpose(vx53(:,p))];
end

k=111;
%*************************************************************************
%FORMANDO A POPULACAO DE INDIVIDUOS (para 2 furos)
%*************************************************************************
for pp = 1:(k-1)
    pop2(pp,1:(n_medido+7)) = [2 transpose(vx1x1(pp)) transpose(vx2y1(pp))
    3 3 0.15 0.03 transpose(vx523(:,pp))];
end
for pp = k:(n_linhas2-(k-1))
    pop2(pp,1:(n_medido+7)) = [2 transpose(vx1x1(pp)) transpose(vx2y1(pp))
    3 3 0.03 0.15 transpose(vx523(:,pp))];
end
for pp = (n_linhas2-(k-2)):n_linhas2
    pop2(pp,1:(n_medido+7)) = [2 transpose(vx1x1(pp)) transpose(vx2y1(pp))
    3 3 0.03 0.03 transpose(vx523(:,pp))];
end
```

```
% pop1
% pop2
pop = vertcat(pop1,pop2);

n_pop = length(pop(1,:)); %NUMERO DE VARIAVEIS

%*************************************************************************
disp('_____________________________________________________________________
_____')
fprintf('\nENCONTRANDO O MÍNIMO DA FUNCAO USANDO O GA\n\n' )
for i = 1:5
    %CRIANDO A ESTRUTURA DE OPCOES DO GA E CALCULANDO O MINIMO
    options = gaoptimset('InitialPopulation',pop, 'PopulationSize',
    (2*p),'Generations',75, 'EliteCount',2,'CrossoverFraction' ,0.8,
    'CrossoverFcn',{@crossoverheuristic,1.25}, 'PlotFcns',@gaplotbestf);
    [popf, fval, reason, output] = ga(@funcional_J3_normaliz2furos, n_pop,
    options)

    %parametros encontrados pelo GA
    num_furo(i) = popf(1,1);
    x_furo1(i) = popf(1,2);
    y_furo1(i) = popf(1,3);
    x_furo2(i) = popf(1,4);
    y_furo2(i) = popf(1,5);
    raio1(i) = popf(1,6);
    raio2(i) = popf(1,7);
    fun_J(i) = fval;
    figure(2)
    hold on
    plot_circulo2(x_furo1(i),y_furo1(i),raio1(i),1)
    hold on

    plot_circulo2(1,3,0.03,2)
    hold on

    plot_circulo2(3,3,0.15,2)
    hold on

    if num_furo >= 1.5
        plot_circulo2(x_furo2(i),y_furo2(i),raio2(i),1)
    end
end

t1 = toc/i; %returns the elapsed time in t.
fprintf('\n\n');
fprintf('tempo de execução: %f segundos\n',t1);
fprintf('\n\n');

%calculando os valores médios e as incertezas com 99.7%
%*************************************************************************
%FURO 1
%*************************************************************************
xmean1 = mean(x_furo1);
xstd1 = std(x_furo1);
xincert1 = 3*std(x_furo1);
ymean1 = mean(y_furo1);
ystd1 = std(y_furo1);
yincert1 = 3*std(y_furo1);
rmean1 = mean(raio1);
rstd1 = std(raio1);
rincert1 = 3*std(raio1);
nfmean1 = mean(num_furo);
```

```
jmean = mean(fun_J);
jstd = std(fun_J);
jincert = 3*std(fun_J);

%*************************************************************************
%NUMERO DE FUROS
%*************************************************************************
if nfmean1 < 1.5
     num_furo = 1;
     %FURO 2
     xmean2 = 0;
     xstd2 = 0;
     xincert2 = 0;
     ymean2 = 0;
     ystd2 = 0;
     yincert2 = 0;
     rmean2 = 0;
     rstd2 = 0;
     rincert2 = 0;
elseif nfmean1 >= 1.5
     num_furo = 2;
     %FURO 2
     xmean2 = mean(x_furo2);
     xstd2 = std(x_furo2);
     xincert2 = 3*std(x_furo2);
     ymean2 = mean(y_furo2);
     ystd2 = std(y_furo2);
     yincert2 = 3*std(y_furo2);
     rmean2 = mean(raio2);
     rstd2 = std(raio2);
     rincert2 = 3*std(raio2);
end

%*************************************************************************
fprintf('\n\nNumero de furos\n')
fprintf('n = %1.3f ',num_furo)
fprintf('\n\nValores encontrados pelo GA (FURO 1)\n')
fprintf('xc = %1.3f ',xmean1)

fprintf('+- %1.3f\t',xincert1)
fprintf('std %1.3f\t',xstd1)
fprintf('\nyc = %1.3f ',ymean1)
fprintf('+- %1.3f\t',yincert1)
fprintf('std %1.3f\t',ystd1)
fprintf('\nr = %1.3f ',rmean1)
fprintf('+- %1.3f\t',rincert1)
fprintf('std %1.3f\t',rstd1)
fprintf('\n\nValores encontrados pelo GA (FURO 2)\n')
fprintf('xc = %1.3f ',xmean2)
fprintf('+- %1.3f\t',xincert2)
fprintf('std %1.3f\t',xstd2)
fprintf('\nyc = %1.3f ',ymean2)
fprintf('+- %1.3f\t',yincert2)
fprintf('std %1.3f\t',ystd2)
fprintf('\nr = %1.3f ',rmean2)
fprintf('+- %1.3f\t',rincert2)
fprintf('std %1.3f\t',rstd2)
fprintf('\n')
fprintf('\nJ = %1.3f ',jmean)
fprintf('+- %1.3f\t',jincert)
fprintf('std %1.3f\t',jstd)
fprintf('\n\n')
%*************************************************************************
```

```
function J = funcional_J3_normaliz2furos(pop)

load('Vx_13r03_33_r15.mat','vx4' ); %dois furos:(1,3) e raio = 0.06; (3,3)
e raio = 0.15

%*************************************************************************
%normalizando o vetor de medidas de diferenca de potencial
%*************************************************************************
%maxTotal = max([max1 … max6 maxmed]) = 4.477197965447841e+006
vx4 = vx4/4.477197965447841e+006;
medido = [transpose(vx4)];

%*************************************************************************
aux = length(pop(1,:));
n = length(pop(:,1));
for w = 1:n
     J(w) = 0.5*sum((medido - pop(w,8:aux)).^2); %a partir do vetor 8 estao
     os valores de potenciais
end
%ENCONTRA A SOMA MINIMA, LOGO, ENCONTRA O DANO...
%*************************************************************************
```

Anexo C

PROGRAMA DE DETECÇÃO DE DANOS UTILIZANDO REDES NEURAIS ARTIFICIAIS

- **Programa de detecção de um furo usando redes neurais artificiais:**

```
%rede neural feedforward backpropagation com 50 neuronios na primeira
%camada intermediária, 4 na segunda camada intermediária e 4 na camada de
%saida. Usa a funcao de ativacao 'tansig' (tang hiperbolica sigmoidal) nas
%camadas 'hidden' e a linear na de saida. A funcao de treinamento eh a
%'traingdx' (Gradient descent with momentum and adaptive learning rate
%backpropagation). Outras possiveis: 'traingd' (Gradient descent
%backpropagation), 'traingda' (Gradient descent with adaptive learning rate
%backpropagation)

clear all
close all
clc

%MONTAGEM DOS DADOS DE ENTRADA (P) E SAIDA (T)
%-------------------------------------------------------------------------
% % - PARA UM FURO
%-------------------------------------------------------------------------
% load('dados25.txt')
% load('dados15.txt')
% load('dados09.txt')
load('dados05.txt') %raio de 0.15
sens = length(dados05(1,:))-4 %4 parametros do furo %dados15, 09, 05
P1 = dados05(:,1:sens)'; %dados15, 09, 05, 25
[nn,kk] = size(dados05) %dados15, 09, 05, 25
for ii = 1:nn
    for jj = (kk-3):kk %(kk-3) == sens + 1
        % T1((jj-25),ii) = [dados25(ii,jj)]; %dados25
        % T1((jj-15),ii) = [dados15(ii,jj)]; %dados15,
        % T1((jj-9),ii) = [dados09(ii,jj)]; %dados09
        T1((jj-5),ii) = [dados05(ii,jj)]; %dados05
```

```
    end
end

% load('dados25_2.txt')
% load('dados15_2.txt')
% load('dados09_2.txt')
load('dados05_2.txt') %raio de 0.05
sens = length(dados05_2(1,:))-4 %4 parametros do furo %dados15, 09, 05
P2 = dados05_2(:,1:sens)'; %dados15, 09, 05, 25
[nn,kk] = size(dados05_2) %dados15, 09, 05, 25
for ii = 1:nn
    for jj = (kk-3):kk %(kk-3) == sens + 1
        % T2((jj-25),ii) = [dados25_2(ii,jj)]; %dados25
        % T2((jj-15),ii) = [dados15_2(ii,jj)]; %dados15,
        % T2((jj-9),ii) = [dados09_2(ii,jj)]; %dados09
        T2((jj-5),ii) = [dados05_2(ii,jj)]; %dados05
    end
end

%--------------------------------------------------------------------------
%concatenando
P = [P1 P2];
T = [T1 T2];
%--------------------------------------------------------------------------

nnOut = length(T(:,1));

tic; %starts a stopwatch timer.

%CRIACAO DA REDE
net = newff(minmax(P),[50 4 nnOut],{'tansig' 'tansig''purelin'},
'traingdx'); %rede neural do tipo feedforward backpropagation

%CONFIGURACAO DOS PARAMETROS E TREINO DA REDE
net.trainParam.goal = 1e-15; %erro desejado
net.trainParam.epochs = 5000;
net.trainParam.lr = 0.05; %taxa de aprendizado
net = train(net,P,T); %treinamento da rede

T %VETOR TARGET (SAIDA DESEJADA)

%SIMULACAO APOS O TREINO
Y = sim(net,P) %simulacao da ANN

%--------------------------------------------------------------------------
% MEDIDO 1 FURO:
%--------------------------------------------------------------------------
% load('dados_med33.txt'); %1 furo na posicao (3,3) e de raio = 0.09
% load('f33r010_25ptos.txt'); %1 furo na posicao (3,3) e de raio = 0.10
% load('f33r010_15ptos.txt'); %1 furo na posicao (3,3) e de raio = 0.10
% load('f33r010_09ptos.txt'); %1 furo na posicao (3,3) e de raio = 0.10
load('f33r010_05ptos.txt'); %1 furo na posicao (3,3) e de raio = 0.10

sens = length(f33r010_05ptos(1,:))-4; %numero de sensores
med = f33r010_05ptos(:,1:sens)';
Y = sim(net,med) %simulacao da ANN
n_furo = Y(1);
x_furo = Y(2);
y_furo = Y(3);
raio = Y(4);

figure(1)
subplot(3,2,1); plot_circulo2(Y(2),Y(3),Y(4),1)
```

```
% hold on
plot_circulo2(3,3,0.10,2)

%-------------------------------------------------------------------------
load('f55r010_05ptos.txt'); %1 furo na posicao (5,5) e de raio = 0.10

sens = length(f55r010_05ptos(1,:))-4; %numero de sensores!!!
med = f55r010_05ptos(:,1:sens)';
Y = sim(net,med) %simulacao da ANN
n_furo = Y(1);
x_furo = Y(2);
y_furo = Y(3);
raio = Y(4);

% figure(2)
subplot(3,2,2); plot_circulo2(Y(2),Y(3),Y(4),1)
plot_circulo2(5,5,0.10,2)

%-------------------------------------------------------------------------

load('f42r010_05ptos.txt'); %1 furo na posicao (4,2) e de raio = 0.10

sens = length(f42r010_05ptos(1,:))-4; %numero de sensores!!!
med = f42r010_05ptos(:,1:sens)';
Y = sim(net,med) %simulacao da ANN
n_furo = Y(1);
x_furo = Y(2);
y_furo = Y(3);
raio = Y(4);

% figure(3)
subplot(3,2,3); plot_circulo2(Y(2),Y(3),Y(4),1)
plot_circulo2(4,2,0.10,2)

%-------------------------------------------------------------------------
load('f24r010_05ptos.txt'); %1 furo na posicao (2,4) e de raio = 0.10

sens = length(f24r010_05ptos(1,:))-4; %numero de sensores!!!
med = f24r010_05ptos(:,1:sens)';
Y = sim(net,med) %simulacao da ANN
n_furo = Y(1);
x_furo = Y(2);
y_furo = Y(3);
raio = Y(4);

% figure(4)
subplot(3,2,4); plot_circulo2(Y(2),Y(3),Y(4),1)
plot_circulo2(2,4,0.10,2)

%-------------------------------------------------------------------------
load('f11r010_05ptos.txt'); %1 furo na posicao (1,1) e de raio = 0.10

sens = length(f11r010_05ptos(1,:))-4; %numero de sensores!!!
med = f11r010_05ptos(:,1:sens)';
Y = sim(net,med) %simulacao da ANN
n_furo = Y(1);
x_furo = Y(2);
y_furo = Y(3);
raio = Y(4);

% figure(5)
subplot(3,2,5); plot_circulo2(Y(2),Y(3),Y(4),1)
plot_circulo2(1,1,0.10,2)
```

```
%-------------------------------------------------------------------------

t1 = toc; %returns the elapsed time in t.

fprintf('\n\n');
fprintf('tempo de execução: %f segundos\n',t1);
fprintf('\n\n');
%-------------------------------------------------------------------------
```

- **Programa de detecção de até dois furos usando redes neurais artificiais:**

```
%rede neural feedforward backpropagation com 50 neuronios na primeira
%camada intermediária, 4 na segunda camada intermediária e 4 na camada de
%saida. Usa a funcao de ativacao 'tansig' (tang hiperbolica sigmoidal) nas
%camadas 'hidden' e a linear na de saida. A funcao de treinamento eh a
%'traingdx' (Gradient descent with momentum and adaptive learning rate
%backpropagation). Outras possiveis: 'traingd' (Gradient descent
%backpropagation), 'traingda' (Gradient descent with adaptive learning rate
%backpropagation)

clear all
close all
clc

%MONTAGEM DOS DADOS DE ENTRADA (P) E SAIDA (T)
%-------------------------------------------------------------------------
% % - PARA UM FURO
%-------------------------------------------------------------------------
load('dados25_2f_4.txt')
% load('dados15.txt')
% load('dados09.txt')
% load('dados05.txt')
sens = length(dados25_2f_4(1,:))-7 %4 parametros do furo %dados15, 09, 05
P0 = dados25_2f_4(:,1:sens)'; %dados15, 09, 05, 25
[nn,kk] = size(dados25_2f_4) %dados15, 09, 05, 25
for ii = 1:nn
    for jj = (kk-6):kk %(kk-3) == sens + 1
        T0((jj-25),ii) = [dados25_2f_4(ii,jj)]; %dados25
        % T0((jj-15),ii) = [dados15(ii,jj)]; %dados15,
        % T0((jj-9),ii) = [dados09(ii,jj)]; %dados09
        % T0((jj-5),ii) = [dados05(ii,jj)]; %dados05
    end
end

load('dados25_2f_5.txt')
sens = length(dados25_2f_5(1,:))-7 %4 parametros do furo %dados15, 09, 05
P5 = dados25_2f_5(:,1:sens)'; %dados15, 09, 05, 25
[nn,kk] = size(dados25_2f_5) %dados15, 09, 05, 25
for ii = 1:nn
    for jj = (kk-6):kk %(kk-3) == sens + 1
        T5((jj-25),ii) = [dados25_2f_5(ii,jj)]; %dados25
        % T5((jj-15),ii) = [dados15_2(ii,jj)]; %dados15,
        % T5((jj-9),ii) = [dados09_2(ii,jj)]; %dados09
        % T5((jj-5),ii) = [dados05_2(ii,jj)]; %dados05
    end
end

%-------------------------------------------------------------------------
% - PARA 1 OU 2 FUROS
```

```
%-------------------------------------------------------------------------
load('dados25_2f.txt')
sens = length(dados25_2f(1,:))-7; %7 parametros dos 2 furos
P1 = dados25_2f(:,1:sens)';
[nn,kk] = size(dados25_2f);
for ii = 1:nn
    for jj = (kk-6):kk %(kk-3) == sens + 1
        T1((jj-25),ii) = [dados25_2f(ii,jj)];
    end
end

load('dados25_2f_2.txt')
sens = length(dados25_2f_2(1,:))-7; %7 parametros dos 2 furos
P2 = dados25_2f_2(:,1:sens)';
[nn,kk] = size(dados25_2f_2);
for ii = 1:nn
    for jj = (kk-6):kk %(kk-3) == sens + 1
        T2((jj-25),ii) = [dados25_2f_2(ii,jj)];
    end
end

load('dados25_2f_3.txt')
sens = length(dados25_2f_3(1,:))-7; %7 parametros dos 2 furos
P3 = dados25_2f_3(:,1:sens)';
[nn,kk] = size(dados25_2f_3);
for ii = 1:nn
    for jj = (kk-6):kk %(kk-3) == sens + 1
        T3((jj-25),ii) = [dados25_2f_3(ii,jj)];
    end
end

load('dados25_2f_32.txt' )
sens = length(dados25_2f_32(1,:))-7; %7 parametros dos 2 furos
P4 = dados25_2f_32(:,1:sens)';
[nn,kk] = size(dados25_2f_32);
for ii = 1:nn
    for jj = (kk-6):kk %(kk-3) == sens + 1
        T4((jj-25),ii) = [dados25_2f_32(ii,jj)];
    end
end

%-------------------------------------------------------------------------
%concatenando
P = [P0 P1 P2 P3 P4 P5];
T = [T0 T1 T2 T3 T4 T5];
%-------------------------------------------------------------------------

nnOut = length(T(:,1));

tic; %starts a stopwatch timer.

%CRIACAO DA REDE
net = newff(minmax(P),[50 4 nnOut],{'tansig' 'tansig' 'purelin'},
'traingda'); %rede neural do tipo feedforward backpropagation

%CONFIGURACAO DOS PARAMETROS E TREINO DA REDE
net.trainParam.goal = 1e-15; %erro desejado
net.trainParam.epochs = 5000;
net.trainParam.lr = 0.005; %taxa de aprendizado
net = train(net,P,T); %treinamento da rede

T %VETOR TARGET (SAIDA DESEJADA)
```

```
%SIMULACAO APOS O TREINO
Y = sim(net,P) %simulacao da ANN

%-------------------------------------------------------------------------
% MEDIDO 1 FURO:
%-------------------------------------------------------------------------
% load('dados_med33.txt'); %1 furo na posicao (3,3) e de raio = 0.09
load('f33r010_25ptos.txt'); %1 furo na posicao (3,3) e de raio = 0.10
% load('f33r010_15ptos.txt'); %1 furo na posicao (3,3) e de raio = 0.10
% load('f33r010_09ptos.txt'); %1 furo na posicao (3,3) e de raio = 0.10
% load('f33r010_05ptos.txt'); %1 furo na posicao (3,3) e de raio = 0.10

sens = length(f33r010_25ptos(1,:))-4; %numero de sensores!!!
med = f33r010_25ptos(:,1:sens)';
Y = sim(net,med) %simulacao da ANN
x_furo = Y(2);
y_furo = Y(3);
raio = Y(4);
%figure(1)
subplot(3,2,1); plot_circulo2(Y(2),Y(3),Y(4),1)
% hold on
plot_circulo2(3,3,0.10,2)

%-------------------------------------------------------------------------
load('f42r010_25ptos.txt'); %1 furo na posicao (4,2) e de raio = 0.10

sens = length(f42r010_25ptos(1,:))-4; %numero de sensores!!!
med = f42r010_25ptos(:,1:sens)';
Y = sim(net,med) %simulacao da ANN
x_furo = Y(2);
y_furo = Y(3);
raio = Y(4);
subplot(3,2,2); plot_circulo2(Y(2),Y(3),Y(4),1)
% hold on
plot_circulo2(4,2,0.10,2)

%-------------------------------------------------------------------------
% MEDIDO 2 FUROS:
%-------------------------------------------------------------------------
load('f13r010_53r010_25ptos.txt' ); %2 furos: (1,3) e r=0.10; (5,3) e
r=0.10
sens = length(f13r010_53r010_25ptos(1,:))-7;
med = f13r010_53r010_25ptos(:,1:sens)';

Y = sim(net,med) %simulacao da ANN
n_furo = Y(1);
x_furo1 = Y(2);
y_furo1 = Y(3);
raio1 = Y(4);
x_furo2 = Y(5);
y_furo2 = Y(6);
raio2 = Y(7);

% figure(1)
subplot(3,2,3); plot_circulo2(Y(2),Y(3),Y(4),1)
% hold on
plot_circulo2(Y(5),Y(6),Y(7),1)

% hold on
plot_circulo2(1,3,0.10,2)
% hold on
plot_circulo2(5,3,0.10,2)
```

```
%-------------------------------------------------------------------------
load('f11r015_33r010_25ptos.txt' ); %2 furos: (1,3) e r=0.10; (5,3) e
r=0.10
sens = length(f11r015_33r010_25ptos(1,:))-7;
med = f11r015_33r010_25ptos(:,1:sens)';

Y = sim(net,med) %simulacao da ANN
n_furo = Y(1);
x_furo1 = Y(2);
y_furo1 = Y(3);
raio1 = Y(4);
x_furo2 = Y(5);
y_furo2 = Y(6);
raio2 = Y(7);

% figure(1)
subplot(3,2,4); plot_circulo2(Y(2),Y(3),Y(4),1)
% hold on
plot_circulo2(Y(5),Y(6),Y(7),1)
% hold on
plot_circulo2(1,1,0.15,2)
% hold on
plot_circulo2(3,3,0.10,2)

%-------------------------------------------------------------------------
load('f55r015_33r010_25ptos.txt' ); %2 furos: (1,3) e r=0.10; (5,3) e
r=0.10
sens = length(f55r015_33r010_25ptos(1,:))-7;
med = f55r015_33r010_25ptos(:,1:sens)';

Y = sim(net,med) %simulacao da ANN
n_furo = Y(1);
x_furo1 = Y(2);
y_furo1 = Y(3);
raio1 = Y(4);
x_furo2 = Y(5);
y_furo2 = Y(6);
raio2 = Y(7);

% figure(1)
subplot(3,2,5); plot_circulo2(Y(2),Y(3),Y(4),1)
% hold on
plot_circulo2(Y(5),Y(6),Y(7),1)
% hold on
plot_circulo2(5,5,0.15,2)
% hold on
plot_circulo2(3,3,0.10,2)

%-------------------------------------------------------------------------

t1 = toc; %returns the elapsed time in t.

fprintf('\n\n');

fprintf('tempo de execução: %f segundos\n',t1);
fprintf('\n\n');
%-------------------------------------------------------------------------
```