Rafael Luís Ferreira de Almeida

Licenciado em Ciências de Engenharia Mecânica

Aplicação de Redes Neuronais Artificiais para Aproximar o Comportamento de Estruturas

Dissertação para obtenção do Grau de Mestre em
Engenharia Mecânica

Orientador: Prof. Doutor João M. Burguete Botelho Cardoso (FCT/UNL)

Júri:

Presidente: Prof. Doutora Raquel Albuquerque Soares Brás de Almeida
Vogais: Prof. Doutor Luís Armando Canhoto Neves
Prof. Doutor João Mário Burguete Botelho Cardoso

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Julho 2012

Copyright

Aplicação de Redes Neuronais Artificiais para Aproximar o Comportamento de Estruturas

Faculdade de Ciências e Tecnologia,

Universidade Nova de Lisboa

Agradecimentos

Agradeço a Deus, pelo dom da vida e pela força em cada jornada.

Ao meu orientador, Professor Doutor João Cardoso, pelos seus conselhos, disponibilidade e conhecimentos transmitidos ao longo desta dissertação.

À minha família, pelo apoio constante ao longo do meu percurso académico, em particular aos meus pais que possibilitaram e incentivaram a minha formação.

Aos meus amigos, por tudo o que fizeram por mim até hoje.

Aos meus colegas de curso, pelas experiências e conhecimentos partilhados ao longo deste percurso.

Resumo

A presente dissertação tem como objetivo desenvolver redes neuronais artificiais em ambiente MATLAB para aproximar a resposta de estruturas de comportamento não linear.

Numa primeira fase utiliza-se uma função não linear para testar a capacidade do MATLAB gerar redes neuronais artificiais com resultados satisfatórios.

Na fase seguinte estuda-se uma treliça de seis barras, analisando-a com o programa de elementos finitos ANSYS e retirando desta forma resultados para poder treinar uma rede e posteriormente testar a sua capacidade de aproximar os resultados aos obtidos pelo programa de elementos finitos. Utiliza-se o método de Monte Carlo para determinar a probabilidade de colapso da estrutura. Para otimização estrutural recorre-se à técnica dos algoritmos genéticos, recorrendo à sua interface gráfica integrada no MATLAB.

Por fim utiliza-se como exemplo uma placa reforçada da estrutura do casco de um navio, analisando-a com o programa de elementos finitos ANSYS e calculando resultados para poder treinar uma rede neuronal. Tratando-se de um problema com oito variáveis, utiliza-se a técnica do hipercubo latino por forma a cobrir o domínio pretendido, mas realizando um menor número de análises, reduzindo desta forma o tempo de cálculo. Utiliza-se no final o método de Monte Carlo para determinar a probabilidade de colapso desta estrutura.

Palavras-chave:

Redes Neuronais Artificiais; Fiabilidade de Estruturas; Método de Monte Carlo; Hipercubo Latino; MATLAB.

Abstract

This dissertation aims to develop artificial neural networks in MATLAB to approximate accurately complex structural response.

As a first step, a nonlinear function is used to test the ability of the MATLAB to develop neural networks with satisfactory results.

In the following step, a six element truss is studied. The finite element program ANSYS is used to obtain structural response in order to train a neural network and later to test their ability to approximate the results generated by the finite element program. Monte Carlo simulation is used to determine the probability of structural failure. For structural optimization, the genetic algorithms technique is used through the MATLAB genetic algorithm toolbox.

Finally stiffened plate structure of a ship hull is used as an example. The finite element program ANSYS is again used to obtain the collapse load needed to train a neural network. Since this is a problem with eight variables, latin hypercube technique is used to cover the required domain, but with a lower number of analyses and so reducing the computation time. Monte Carlo simulation is used to determine the probability of structural failure.

Keywords:
Artificial Neural Networks, Structural Reliability, Monte Carlo Simulation, Latin Hypercube, MATLAB.

Índice de Matérias

Índice de Figuras

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Índice de Tabelas

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Simbologia e Notações

Latim

A	Área de secção de cada barra
a	Distância entre reforços transversais e comprimento da placa reforçada; Soma ponderada das entradas da rede neuronal
b	bias; Largura da placa reforçada
Cov	Coeficiente de variação
E	Módulo de Young
$f(a)$	Função de ativação ou de transferência
f_a	Tensão atuante na placa reforçada
f_R	Função densidade de probabilidade da resistência da estrutura
f_{RS}	Função densidade de probabilidade conjunta
f_S	Função densidade de probabilidade da ação
$F_X(x)$	Função de distribuição acumulada
$f_X(x)$	Função densidade de probabilidade
F_{xj}^{-1}	Inverso da função de distribuição cumulativa para a variável de destino j
$g()$	Função de estado limite
$g(u)$	Função de estado limite não linear
$g'(u)$	Função linear
g_i	Função e falha da barra i
H	Matriz Hessiana
i	Índice do neurónio, número da barra e Ordem do momento
I	Matriz identidade
j	Variável de destino
$J(w)$	Matriz Jacobiana
K	Número de variáveis aleatórias
$\overline{\boldsymbol{L}}$	Matriz triangular inferior
m	Identificação do número da camada
M	Número de camadas
m_i	Momento de ordem i
M_{sw}	Momento águas paradas

M_{wv}	Momento induzido ondulação
n	*N*úmero de variáveis básicas do problema
N	Número total de simulações e número de realizações
n_c	Número de simulações em que ocorre colapso
N_m	Número de neurónios da camada m
nx	Deslocamento do nó 4
o_{ij}	Valores reais da função
P	Número de conjuntos e carga aplicada no nó 3 do exemplo da placa reforçada
$\boldsymbol{P}$	Matriz, $N \times K$
$P()$	Probabilidade
p_c	Probabilidade de colapso
p_{ij}	Elementos da matriz P
R	Resistência da estrutura
$R[x, y, z]$	Restrições rotacionais
S	Ação
$\boldsymbol{S}$	Matriz que representa o plano de amostragem
s^0	Camada entrada
s^1	Camada intermédia
s^2	Camada saída
S_{ij}	Elementos da matriz S
s_{ij}	Valores obtidos pela simulação da rede neuronal
$T[x, y, z]$	Restrições de translação
t_1	Espessura placa
t_2	Espessura alma reforço
t_3	Espessura banzo reforço
Ti	Tensão na barra i
u^*	Ponto de dimensionamento
$Var[X]$	Variância
w_i	Peso da ligação
$w_{ij}^{(m)}$	Peso da ligação entre o neurónio j da camada m e o neurónio i da camada (m+1)
w_{oc}	Empeno longitudinal
w_{os}	Empeno reforço
w_p	Empeno da placa
X	Vetor das variáveis aleatórias ou das variáveis básicas do problema
x_i^d	Saída desejada
$\hat{x}_{ij}$	Dados de entrada para uma computação determinística

x_i	Valores de entrada
x_i^m	Saída do neurónio *i* da camada *m*
y	Saída do neurónio
Z	Margem de segurança

Grego

α	Vetor perpendicular à função estado limite no ponto de dimensionamento
$\alpha_i^{(M)}$	Sinal de erro generalizado
β	Índice de fiabilidade
$\Delta w_{ij}^{(m)}$	Variação feita no peso $w_{ij}^{(m)}$
μ_{g}	Valor médio da função estado limite
μ_{R}	Valor médio das resistências
μ_{S}	Valor médio das ações
μ_x	Valor médio de uma variável aleatória contínua
ν	Coeficiente de Poisson
ρ	Densidade do material; Rácio entre os custos totais e os custos de construção
σ_c	Tensão de cedência
σ_g	Desvio padrão da função estado limite
σ_R	Desvio padrão das resistências
σ_S	Desvio padrão das ações
Φ	Função distribuição normal reduzida
Φ^{-1}	Inverso da função distribuição normal reduzida
ε	Erro relativo
η	Coeficiente de aprendizagem

Abreviaturas

ADELINE	*ADAptive LINear Element*
CDF	Funções distribuição acumulada
FORM	*First Order Reliability Method*
FRS	Funções de Resposta em Superfície
JCSS	*Joint Committee on Structural Safety*
LMS	*Least Mean Square Rule*

MEF	Método dos Elementos Finitos
PDF	Funções densidade de probabilidade
RNA	Redes Neuronais Artificiais
SORM	*Second Order Reliability Method*

Capítulo1

Introdução

A segurança é o principal aspeto quando se projetam estruturas. Desta forma, métodos para estudar a fiabilidade de estruturas têm vindo a ser desenvolvidos ao longo do tempo [1].

Ao dimensionar uma estrutura não é possível garantir que esta seja absolutamente segura, dada a aleatoriedade das variáveis envolvidas, o que acarreta sempre um risco para o projeto. A teoria da fiabilidade de estruturas pretende estudar as incertezas envolvidas no projeto, de maneira a obter valores aceitáveis da probabilidade de colapso (p_c) da estrutura, ou seja, a probabilidade da estrutura não ter um desempenho satisfatório ao longo da sua vida útil [1]. A análise probabilística está hoje na base da regulamentação utilizada no estudo da segurança estrutural, nomeadamente dos documentos "JCSS – Probabilistic Model Code" [2] e "Eurocódigo 0" [3].

De entre os métodos existentes mais usados em fiabilidade de estruturas, recorre-se nesta dissertação ao método de simulação de Monte Carlo para calcular a probabilidade de colapso das estruturas utilizadas como exemplos de estudo.

O comportamento estrutural pode ser determinado com grande precisão através de simulações efetuadas usando modelos numéricos, sendo atualmente habituais os modelos de elementos finitos. O método dos elementos finitos (MEF), utilizado quando o problema em análise é demasiado complexo para ser resolvido de forma satisfatória pelos métodos clássicos de análise, é um método numérico que aproxima a solução de problemas com condições de fronteira descritas por equações diferenciais [4, 5].

No MEF o domínio é subdividido em regiões, ou elementos finitos, ligadas entre si através de pontos nodais, denominados nós. Ao conjunto dos elementos finitos e dos nós, dá-se

o nome de malha de elementos finitos. Através da soma da contribuição de cada elemento, chega-se a um sistema total de equações que depois de resolvidas permitem conhecer os valores das incógnitas nos pontos nodais [4, 5].

Neste trabalho o MEF é aplicado através do programa ANSYS. O ANSYS é um programa que existe há quarenta anos e foi pioneiro na aplicação de métodos de elementos finitos [6].

O programa é utlizado na resolução de problemas mecânicos como sejam: análises estáticas e dinâmicas de estruturas, análise de transferência de calor, análise de problemas acústicos e também de eletromagnetismo [6].

O programa está dividido em três ferramentas principais chamadas: pré-processador (*Preprocessor*), solução (*Solution*) e pós-processador (*Postprocessor*) [6]. No pré-processador é criado um modelo numérico da estrutura através da definição do tipo de elemento estrutural (viga, placa, casca, etc.), da definição das constantes características do elemento e do tipo de material empregue. São ainda numerados os nós e os elementos e definidas as forças que atuam na estrutura e as suas condições de apoio bem como o tipo de análise escolhido. Na ferramenta solução, são efetuados os cálculos e finalmente a ferramenta de pós-processamento é utilizada para a apresentação dos resultados da análise da fase de solução.

Contudo, para estruturas de grande dimensão e de comportamento não linear e quando se pretendem aplicar metodologias em que é necessário um grande número de análises para valores semelhantes dos parâmetros do modelo, como acontece no método de Monte Carlo, várias técnicas de obtenção da resposta, sem recorrer a modelos de elementos finitos, têm sido utilizadas. Estas técnicas revelam-se fiáveis, económicas e mais rápidas que os métodos convencionais que utilizam elementos finitos. Entre elas destacam-se as Redes Neuronais Artificiais (RNA) e as funções de resposta em superfície (FRS).

As RNA, descritas em pormenor na secção 2.2 da presente dissertação, são utilizadas neste trabalho para aproximar a resposta de estruturas de comportamento não linear. A sua utilização é sugerida por Cardoso et al. [7] que apresenta uma solução para dois dos exemplos apresentados usando RNA. Na bibliografia encontram-se outros exemplos que também serviram de motivação e suporte a esta dissertação, dos quais se destacam [1, 4, 8, 9, 10].

1.1. Objetivo

O grande objetivo desta dissertação é testar a capacidade das redes neuronais do MATLAB para aproximar a resposta de estruturas de comportamento não linear. Os resultados das RNA são comparados com os obtidos com análises por elementos finitos realizadas com o programa ANSYS, calculando-se desta forma o erro associado à resposta da rede.

Outro objetivo consiste em aplicar as RNA na determinação da fiabilidade de estruturas, com os resultados obtidos para essa fiabilidade a serem comparados com os obtidos por outros métodos usados nesse domínio, como o FORM (*First Order Reliability Method*).

Finalmente será investigada a possibilidade de realizar otimização estrutural, com constrangimentos de fiabilidade, utilizando algoritmos genéticos e as RNA criadas em ambiente MATLAB.

A investigação realizada nesta dissertação insere-se no âmbito projeto PTDC/ECM/115932/2009 – "Métodos adaptativos para a análise de fiabilidade de estruturas complexas".

1.2. Estrutura da Dissertação

A dissertação apresenta-se organizada em sete capítulos.

Neste primeiro capítulo, introduz-se o tema da dissertação e apresenta-se a sua motivação. São também indicados os objetivos e uma breve descrição da estrutura da dissertação.

No segundo capítulo são apresentados os fundamentos teóricos que suportam as matérias estudadas nos capítulos seguintes. Apresentam-se as noções de segurança estrutural, fiabilidade de estruturas, do método de Monte Carlo, de otimização estrutural e de algoritmos genéticos. Abordam-se as redes neuronais artificiais, descrevendo os fundamentos biológicos, comparando com o modelo artificial e apresentando uma revisão histórica da sua evolução. São descritos tipos de rede e de treino de redes neuronais artificiais.

No terceiro capítulo inicia-se a componente prática desta dissertação, apresentando-se as interfaces gráficas do MATLAB usadas para desenvolver redes neuronais artificiais. Recorrendo a uma função não linear, são descritos os procedimentos para utilizar cada uma das

interfaces gráficas e apresentam-se os resultados, por forma a verificar a capacidade do MATLAB no desenvolvimento de redes neuronais artificiais.

No quarto capítulo utiliza-se como exemplo uma estrutura de comportamento não linear, designada por treliça de seis barras, a qual é analisada pelo programa ANSYS, sendo os resultados obtidos usados para treinar e simular uma rede desenvolvida em MATLAB. Este exemplo é ainda aproveitado para realizar otimização estrutural, com constrangimentos de fiabilidades, recorrendo a algoritmos genéticos.

No quinto capítulo, um painel reforçado pertencente ao casco de um navio é utilizado como exemplo, sendo analisado pelo programa ANSYS. Os resultados obtidos são usados para treinar e simular uma rede desenvolvida em MATLAB. O método de Monte Carlo é empregue em conjunto com a rede neuronal para obter a probabilidade de colapso da placa. Descrevem-se os procedimentos seguidos e apresentam-se os resultados obtidos.

No sexto e último capítulo apresentam-se as conclusões gerais retiradas deste trabalho, bem como as possibilidades de aplicação e desenvolvimento deste tema no futuro.

Capítulo2

Fundamentos Teóricos

O presente capítulo tem como objetivo introduzir o conhecimento teórico necessário para o desenvolvimento do trabalho, clarificando os conceitos que serão utilizados nos capítulos seguintes.

2.1. Segurança Estrutural

A sociedade espera que as suas habitações, locais de trabalho, pontes, monumentos e, em geral todo o tipo de estruturas utilizadas sejam seguras e que o colapso das mesmas seja um acontecimento muito raro. No entanto existe a consciencialização de que muitas causas podem contribuir para a redução da vida útil de uma estrutura, como a falta de qualidade na construção, a utilização de normas e de critérios de projeto errados, o agravamento das cargas a que a estrutura está sujeita, entre outros [11].

A avaliação da segurança estrutural tem vindo a evoluir ao longo dos tempos, de uma formulação empírica, em que muitas das decisões dependiam essencialmente da experiência pessoal, intuição e sentido crítico, para a aplicação de métodos de fiabilidade estrutural, com o objetivo de efetuar uma avaliação mais rigorosa e consistente dessa segurança [12, 13].

Durante o processo de construção e utilização de uma estrutura esta apresenta um comportamento que depende de um conjunto de fatores que não podem ser controlados de uma forma absoluta [12, 13]. A incapacidade de controlar todos os fatores gera incertezas associadas ao problema da avaliação da segurança. Uma forma de contornar esta dificuldade consiste em

considerar como não determinísticos ou aleatórios os parâmetros de projeto em relação aos quais a incerteza é maior.

Desta forma surgiu a necessidade de desenvolver um conceito de segurança sob uma perspetiva probabilística [11], ponderando a variabilidade dos fatores intervenientes no problema. A segurança é, então, avaliada de acordo com o conceito de probabilidade de colapso para os eventuais estados limites, ou seja, a probabilidade da estrutura não ter um desempenho satisfatório. Neste contexto, surgiu o conceito de fiabilidade estrutural, definida como a capacidade de uma estrutura ou de um membro estrutural, cumprir os requisitos especificados para a qual foi concebido, durante a sua vida útil [12]. De acordo com a atual regulamentação, entende-se por vida útil de uma estrutura, o período de tempo para o qual uma estrutura ou parte dela deve ser utilizado para os fins pretendidos com a manutenção prevista, mas sem que sejam necessárias grandes reparações [12].

A fiabilidade estrutural pode ser então encarada como uma ferramenta para a avaliação da probabilidade de colapso, tendo como objetivo o tratamento das incertezas envolvidas no projeto de forma a garantir um valor aceitável desta probabilidade para todas as estruturas [14].

2.1.1. Métodos de Análise de Segurança Estrutural

Na avaliação da segurança de uma estrutura existem inúmeras incertezas associadas às variáveis intervenientes na caracterização de ações e resistências, sendo que muitas destas não são conhecidas na sua totalidade. Assim sendo, os métodos de análise de segurança pretendem estimar um valor para essas incertezas tão próximo do real quanto possível.

A abordagem das variáveis é diferente, consoante o método de análise escolhido. Esta pode ser desde puramente determinística a puramente probabilística, sendo os resultados mais relevantes para a atual engenharia, os obtidos pelos métodos semi-probabilístico e probabilístico.

Geralmente os métodos para verificar a segurança aos estados limites podem ser divididos em quatro níveis [12, 13], mencionados com níveis de complexidade e confiança crescente:

Nível 0 (Determinísticos)

Corresponde a análises puramente determinísticas, sendo as variáveis envolvidas, consideradas através de valores estritamente determinísticos. As incertezas são ponderadas através de um coeficiente global de segurança, que visa representar simultaneamente a

variabilidade das ações e das resistências. Não tendo uma base probabilística, baseando-se na experiência do engenheiro face ao problema em causa, este método é fundamentalmente empírico.

Nível 1 (Semi-probabilísticos)

Baseiam-se na utilização de coeficientes parciais de segurança, obtidos através da utilização de métodos probabilísticos de nível superior. Neste caso a variabilidade das ações bem como das resistências, é considerada através de valores médios ou característicos, afetados de coeficientes de segurança. Torna-se assim possível substituir as funções de distribuição por valores determinísticos possíveis de serem utilizados de uma forma simples.

As normas atuais [2, 3] baseiam-se neste método, onde os coeficientes são utilizados como forma de modelar as incertezas de um modo mais acessível.

Nível 2 (Probabilísticos simplificados)

As variáveis são definidas pela sua média e desvio padrão, sendo a relação entre as variáveis descrita pela covariância. A segurança é realizada com base na probabilidade de uma função de estado limite não ser violada. Nestes métodos, a medida de segurança utilizada denomina-se índice de fiabilidade (β), que está diretamente relacionado com a probabilidade de colapso da estrutura [2, 3].

Nível 3 (Puramente probabilísticos)

Estes métodos têm em conta a distribuição conjunta de todas as variáveis. As variáveis aleatórias são definidas por distribuições estatísticas com base em observações, tornando este tipo de análise de difícil implementação devido à complexa informação e à necessidade de recorrer a métodos computacionais pesados, isto é, envolvendo cálculos demorados.

2.1.2. Incertezas

Esta secção pretende clarificar as principais fontes de incertezas que condicionam a avaliação do comportamento de uma estrutura, por forma a entender quais os parâmetros que mais influenciam a sua probabilidade de colapso [12].

As fontes de incerteza podem ser agrupadas da seguinte forma [12, 13]:

Incerteza devida a fatores humanos

Este tipo de incerteza deve-se ao envolvimento humano nas várias etapas de elaboração de uma estrutura, desde o processo de documentação, ao dimensionamento, construção e utilização. Dada a natureza deste tipo de incerteza, o conhecimento sobre ela é limitado, sendo por isso difícil de quantificar.

Incerteza física

Deriva da imprevisibilidade do carregamento de uma estrutura, bem como da natureza incerta das propriedades dos materiais e geometria dos elementos. Para controlar este tipo de incerteza é necessário obter o máximo de informação possível sobre as variáveis envolvidas.

Incerteza na modelação

São as incertezas acopladas às simplificações teóricas feitas durante o processo de dimensionamento referentes ao comportamento dos materiais, aplicação das ações e sua resposta.

Incerteza estatística

Resulta da restrição da quantidade de dados disponíveis para caracterizar o modelo probabilístico. Este tipo de incerteza pode ser minimizado obtendo um maior número de dados e utilizando técnicas de inferência estatística como por exemplo, utilização de funções de distribuição de probabilidade.

2.1.3. Conceitos Probabilísticos

Os métodos probabilísticos de nível 2 e 3 para a análise de segurança estrutural recaem essencialmente na estimativa da probabilidade de colapso de uma estrutura, tendo em conta as suas condições reais de funcionamento e utilizando técnicas baseadas na teoria da fiabilidade estrutural [13].

Em fiabilidade estrutural utiliza-se o conceito de probabilidade condicionada, associado à interpretação Bayesiana, onde a probabilidade de dado evento é formulada como o grau de confiança na ocorrência do evento. No âmbito da fiabilidade estrutural, a probabilidade de dado acontecimento ocorrer pode ser interpretada como o grau de confiança que o engenheiro tem perante a realização do evento.

Variáveis aleatórias

Na análise de segurança estrutural, o conjunto de variáveis inclui variáveis determinísticas e aleatórias, podendo as últimas ser do tipo discreto ou contínuo. As variáveis aleatórias contínuas podem tomar qualquer valor dentro de um intervalo, ditado pela função de distribuição associada e pelas medidas estatísticas, das quais, a média e desvio padrão são as mais utilizadas em engenharia de estruturas [12].

Normalmente é impossível remover as incertezas associadas a um determinado parâmetro físico. No entanto, os métodos da teoria da probabilidade tornam possível formular racionalmente, através da estatística, essa incerteza. Por isso é fundamental na aplicação desses métodos decidir quais as variáveis que vão ser consideradas determinísticas e quais serão tomadas como aleatórias.

O desempenho de um sistema estrutural é então determinado com base num modelo dependente dessas variáveis básicas aleatórias que são definidas como os parâmetros que consideram a incerteza no modelo adotado, caracterizando o comportamento da estrutura. Podem ser representadas por um vetor, X designado vetor das variáveis aleatórias ou das variáveis básicas do problema, sendo estas as que têm associada uma distribuição de probabilidade associada, com parâmetros definidos. O vetor X pode ser representado da seguinte forma:

$$X = [X_1, X_2, \dots, X_n] \tag{2.1}$$

Neste vetor *n* representa o número de variáveis básicas do problema, que podem ser parâmetros de carga, resistência, geométricos ou propriedades dos materiais. A relação entre as diferentes variáveis aleatórias pode ser considerada como dependente ou independente [12].

As variáveis aleatórias são caracterizadas por uma determinada tendência comportamental que é obtida através de observações prévias ou modelações teóricas. Quando existe um largo número de observações e registos, é possível obter um diagrama de frequência denominado histograma, através do qual é possível escolher qual o tipo de distribuição de probabilidade que melhor se adequa e ajustar os seus parâmetros. Existem vários tipos de distribuições probabilísticas, sendo utilizadas geralmente em fiabilidade de estruturas as distribuições normal, lognormal, gumbel, etc. Estas distribuições são caracterizadas por funções densidade de probabilidade (PDF) e por funções distribuição acumulada (CDF) específicas.

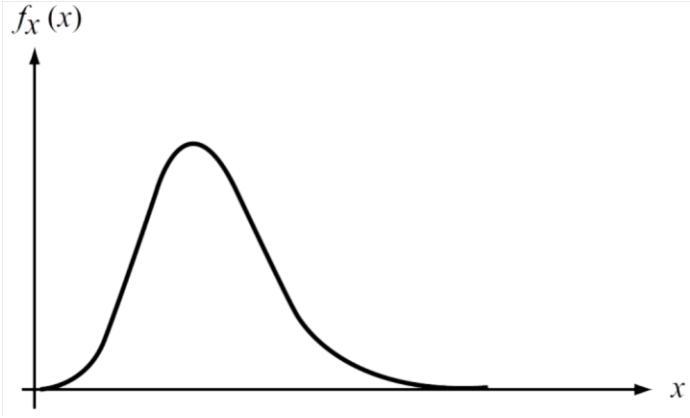


Figura 2. 1 – Função densidade de probabilidade (PDF) [12]

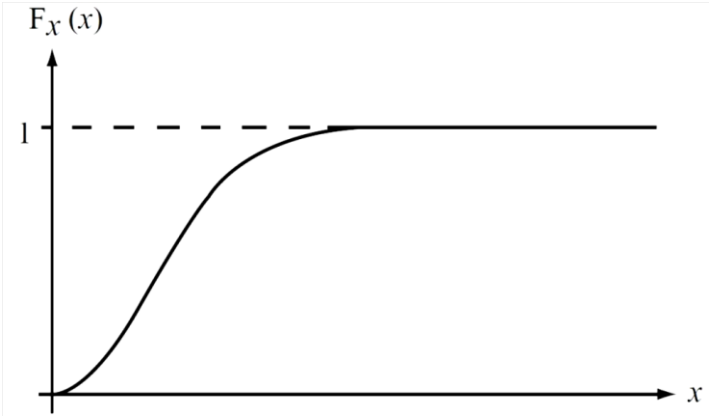


Figura 2. 2 – Função distribuição acumulada [12]

Uma vez escolhida a função distribuição mais adequada e depois de ajustar os seus parâmetros, pode considerar-se que o comportamento de uma variável aleatória é descrito pela função densidade de probabilidade, $f_x(x)$, ou pela função distribuição acumulada, $F_x(x)$,

ilustradas nas figuras 2.1 e 2.2 respetivamente. Esta última define a probabilidade *P*(), que a variável *X* tem de ser menor ou igual a certo valor *x*, expressa por:

$$F_x(x) = P(X \leq x) = \int_{-\infty}^{x} f_x(x)\, dx \tag{2.2}$$

Se $F_X(x)$ for contínua a probabilidade de *X* estar entre os dois valores *a* e *b* é dada por:

$$F_x(b) - F_x(a) = \int_{a}^{b} f_x(x)\, dx \tag{2.3}$$

A função de distribuição acumulada obedece a algumas propriedades, tais como [12]:

- $0 \leq F_X(x) \leq 1$;
- $F(+\infty) = 1$;
- $F(-\infty) = 0$;
- Apresenta um desenvolvimento monótono, não-decrescente;
- A probabilidade de uma variável aleatória ter um determinado valor concreto é zero.

Para os casos em que *X* é contínuo e a primeira derivada existe, pode ainda escrever-se:

$$f_x(x) = \frac{dF_x(x)}{dx} \tag{2.4}$$

Medidas descritivas

As variáveis aleatórias são geralmente definidas pela forma da sua distribuição e por alguns parâmetros, também conhecidos como momentos. Os parâmetros mais utilizados para caracterizar as variáveis são a média e o desvio padrão ou variância (o quadrado do desvio padrão). A média descreve a tendência central da distribuição, enquanto que o desvio padrão é a medida de dispersão em torno do valor médio [11].

Um momento (m_i) de ordem *i* de uma variável aleatória contínua é definida por:

$$m_i = \int_{-\infty}^{+\infty} x^i f_x(x) dx \tag{2.5}$$

O valor médio de uma variável aleatória contínua é definido de acordo com o primeiro momento (i=1):

$$\mu_x = \int_{-\infty}^{+\infty} x\, f_x(x) dx \tag{2.6}$$

A variância ($Var[X]$) é descrita pelo segundo momento central de uma variável contínua e é definida da seguinte forma:

$$\sigma^2 = Var[X] = \int_{-\infty}^{+\infty} (x - \mu_x)^2\, f_x(x) dx \tag{2.7}$$

O coeficiente de variação (*Cov*) é definido pelo quociente entre o desvio padrão e a média:

$$Cov[X] = \frac{\sigma_x}{\mu_x} \tag{2.8}$$

O coeficiente de variação fornece uma medida adimensional da dispersão de uma variável aleatória.

Distribuições de probabilidade

Na avaliação da fiabilidade estrutural, vários tipos de distribuições de probabilidades são usadas para modelar os parâmetros de dimensionamento ou variáveis [12].

A seleção da função de distribuição é uma parte essencial da caracterização probabilística dos sistemas estruturais, dependendo de:

- Natureza do problema;
- Pressupostos subjacentes associados à distribuição;
- Conveniência e simplicidade oferecida pela distribuição, para posterior cálculo.

A tabela 2.1 ilustra as funções distribuição de probabilidade utilizadas ao longo da presente dissertação.

Tabela 2. 1 – Funções de distribuição de probabilidade [11, 12]

Distribuição	Função de Distribuição	Parâmetros	Momentos
Normal	$f_x(x) = \frac{1}{\sigma\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$	μ σ > 0	μ σ
Log-Normal	$f_x(x) = \frac{1}{(x-\varepsilon)\zeta\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{ln(x-\varepsilon)-\lambda}{\zeta}\right)^2\right)$	λ ζ > 0 ε	$\mu = \varepsilon + exp\left(\lambda + \frac{\zeta^2}{2}\right)$ $\sigma = exp\left(\lambda + \frac{\zeta^2}{2}\right)\sqrt{exp(\zeta^2)-1}$
Gumbel	$f_x(x) = e^{\frac{(\alpha-x)}{\zeta} - e^{\frac{(\alpha-x)}{\zeta}}}$	α > 0 ζ > 0	$\mu = \alpha + \gamma\zeta$ $\sigma = \frac{\pi}{\sqrt{6}}\zeta$ Onde γ é a constante de Euler

2.1.4. Estados Limites

A segurança estrutural tem como objetivo encontrar o nível de risco apropriado, bem como um adequado funcionamento da estrutura ao longo da sua vida útil, com o mínimo custo possível.

A avaliação do desempenho de uma estrutura é feito com recurso ao conceito de estado limite, estado que define a condição além da qual uma estrutura não satisfaz os critérios de funcionalidade exigidos. O estado limite pode ser descrito com o auxílio de uma função na forma:

$$Z = g(X) \tag{2.9}$$

onde *X* é o vetor representativo das *n* variáveis básicas do problema, *Z* representa a margem de segurança e *g*() é a função estado limite.

O estado limite define a fronteira entre as regiões de segurança e de rotura da estrutura. Essa fronteira é representada pela função estado limite:

$$Z = g(X) = 0 \tag{2.10}$$

Sendo o estado de rotura dado por,

$$Z = g(X) \leq 0 \tag{2.11}$$

A função estado limite $g(X)$encontra-se representada na figura 2.3, para uma função de duas variáveis básicas, X_1, X_2.

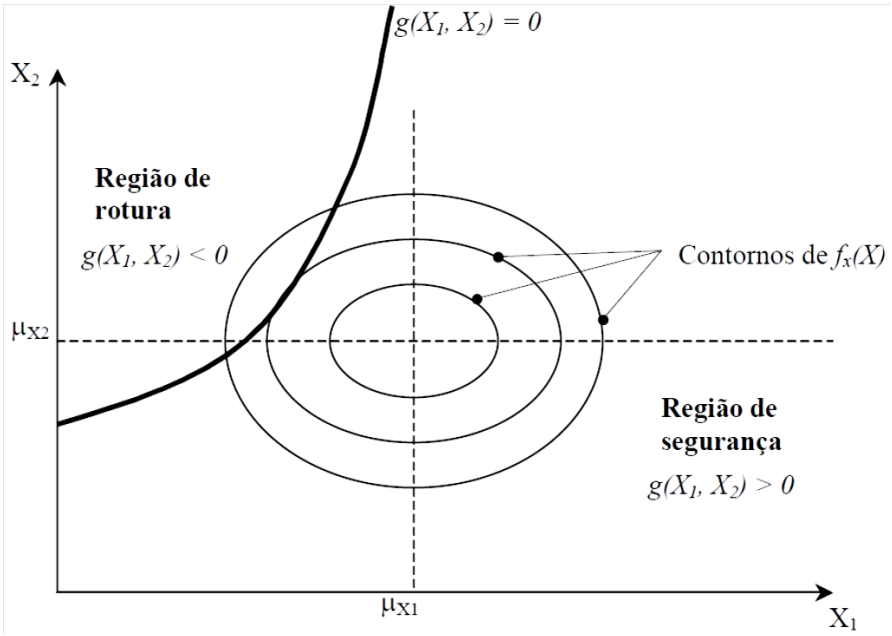


Figura 2. 3 - Representação genérica da função *g*(*X*) e da fronteira entre a região de segurança e a região de rotura [12]

Os estados limites são vulgarmente divididos em estados limites últimos e estados limites de utilização. Os primeiros correspondem a situações de colapso da estrutura em que a segurança de pessoas e equipamento ficam ameaçadas devido a danos estruturais. Os segundos consistem na perda de capacidade da estrutura servir os fins para que foi concebida, devido a danos de origem funcional ou estética [12].

2.1.5. Níveis de Segurança

Os requisitos para a segurança estrutural são expressos através de valores mínimos do índice de fiabilidade (β) ou máximos da probabilidade de colapso, designada por p_c. Os documentos "JCSS – Probabilistic Model Code" [2] e o "Eurocódigo 0" [3], propõem valores recomendados para estes indicadores.

Estes valores são baseados numa otimização de procedimentos, partindo do pressuposto que a maioria das obras de engenharia segue políticas de manutenção razoáveis e sistematizadas [11].

O "Eurocódigo 0 " sugere a adoção de três níveis, apelidados de classes de consequências, por forma a simplificar a escolha do grau de fiabilidade mínimo em função da frequência de utilização e do modo como a estrutura atinge o colapso. A definição de classes de consequências é apresentada na tabela 2.2.

Tabela 2. 2 – Definição da classe de consequências [3]

Classe de consequência	Descrição	Exemplos de edifícios e de obras de engenharia civil
CC1	Consequência baixa em termos de perdas de vidas humanas; ou consequências económicas, sociais ou ambientais pouco importantes ou desprezáveis.	Edifícios agrícolas normalmente não ocupados permanentemente por pessoas (por exemplo, armazéns e estufas).
CC2	Consequência média em termos de perdas de vidas humanas; ou consequências económicas, sociais ou ambientais medianamente importantes.	Edifícios de habitação e de escritórios, edifícios públicos em que as consequências de colapso são médias (por exemplo, um edifício de escritórios).
CC3	Consequência elevada em termos de perdas de vidas humanas; ou consequências económicas, sociais ou ambientais muito importantes.	Bancadas, edifícios públicos em que as consequências de colapso são elevadas (por exemplo, uma sala de concertos).

A tabela 2.3 apresenta os valores mínimos recomendados para o índice de fiabilidade, para o estado limite último, para períodos de retorno de 1 e de 50 anos.

Tabela 2. 3 – Valores mínimos recomendados para o índice de fiabilidade β, para estado limite último. Adaptado de [3]

Classe de consequência	Período de referência de 1 ano	Período de referência de 50 anos
CC1	$\beta = 4,2$	$\beta = 3,3$
CC2	$\beta = 4,7$	$\beta = 3,8$
CC3	$\beta = 5,2$	$\beta = 4,3$

Nota: As classes de fiabilidade dos elementos estruturais acima de CC3, não são consideradas nesta tabela, pois cada um desses elementos exige-se um estatuto específico.

O código modelo "JCSS – Probabilistic Model Code" [2] cria um rácio ρ que é definido como o rácio entre os custos totais e os custos de construção. Os custos totais resultam da soma dos custos de construção e dos custos que a falha da estrutura pode causar [11]. Definindo desta forma as seguintes classes de consequências:

Classe 1: Consequências reduzidas ($\rho < 2$)

Apresenta baixas consequências relativamente a perdas humanas e consequências económicas, sociais ou ambientais pouco importantes ou desprezíveis. Esta classe abrange essencialmente edifícios agrícolas normalmente não ocupados permanentemente por pessoas.

Classe 2: Consequências moderadas ($2 < \rho < 5$)

Está associada a consequências médias a nível de perdas humanas e consequências económicas, sociais ou ambientais medianamente importantes. Esta classe abrange essencialmente edifícios de habitação, escritórios e indústrias em que as consequências de colapso são médias.

Classe 3: Consequências graves ($\rho > 5$)

Caracteriza-se por consequências graves a nível de perdas humanas e consequências económicas, sociais ou ambientais muito significativas. Esta classe abrange os hospitais, grandes pontes e salas de espetáculos, nos quais as consequências de colapso são elevadas.

Estes níveis de fiabilidade estão ajustados de acordo com o tipo de rotura da estrutura, conduzindo assim a valores com maior rigor, na medida em que é espectável que a uma rotura dúctil corresponda um menor valor de índice de fiabilidade do que a uma rotura frágil, devido à impossibilidade de evacuar o edifício, no caso desta última

A tabela 2.4 apresenta os valores mínimos recomendados para o índice de fiabilidade, para o estado limite último.

Tabela 2. 4 - Valores mínimos recomendados para o índice de fiabilidade β, para estado limite último, num período de referência de um ano. Adaptado de [2]

Custos associados a medidas de segurança	Consequências reduzidas	Consequências moderadas	Consequências graves
Elevado	$\beta = 3,1$ ($p_c=10^{-3}$)	$\beta = 3,3$ ($p_c=5 \times 10^{-4}$)	$\beta = 3,7$ ($p_c=10^{-4}$)
Médio	$\beta = 3,7$ ($p_c=10^{-4}$)	$\beta = 4,2$ ($p_c=10^{-5}$)	$\beta = 4,4$ ($p_c=5 \times 10^{-6}$)
Reduzido	$\beta = 4,2$ ($p_c=10^{-5}$)	$\beta = 4,4$ ($p_c=5 \times 10^{-6}$)	$\beta = 4,7$ ($p_c=10^{-6}$)

Em situações de estruturas especiais como, por exemplo, barragens ou centrais nucleares, é conveniente realizar previamente um estudo detalhado dos custos/benefícios, visto que uma eventual falha na estrutura pode traduzir-se em consequências extremas.

2.1.6. Problema Fundamental da Fiabilidade Estrutural

A teoria da fiabilidade estrutural tem como objetivo o cálculo da probabilidade de colapso ou equivalentemente a determinação do índice de fiabilidade estrutural, β [12].

Uma estrutura é considerada fiável se a probabilidade de colapso não exceder o limite previamente definido, em relação ao estado limite em avaliação. O índice de fiabilidade é uma medida quantitativa da fiabilidade, e não pode ser interpretada como uma propriedade física da estrutura, mas sim, como uma medida da sua qualidade global [12].

Por forma a simplificar o problema, dividem-se as variáveis em dois grupos: aquelas que definem a ação (S) e aquelas que definem a resistência da estrutura (R). A verificação de segurança de uma estrutura implica $S<R$, caso contrário ocorre colapso, como se mostra na zona sombreada figura 2.4.

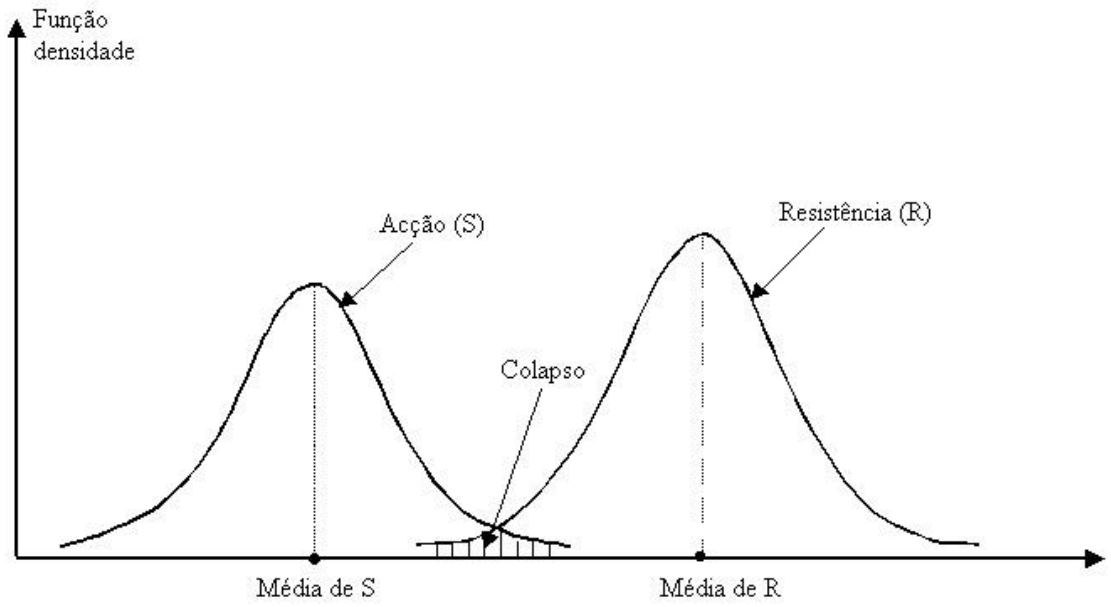


Figura 2. 4 - Funções densidade de probabilidade da ação e resistência

A função que define o estado limite pode ser escrita como:

$$Z = g(X) = R - S \tag{2.12}$$

Em que Z permite quantificar a margem de segurança da estrutura [13]. Em que a superfície limite, que separa a segurança do colapso, é definida por:

$$g(X) = R - S = 0 \tag{2.13}$$

A probabilidade de colapso p_c pode ser descrita como:

$$p_c = P(g(X) \leq 0) = P(R - S \leq 0) \tag{2.14}$$

Na figura 2.5 são apresentadas as funções densidade f_R e f_S para as variáveis R e S, respetivamente, em conjunto com a função densidade de probabilidade conjunta f_{RS}.

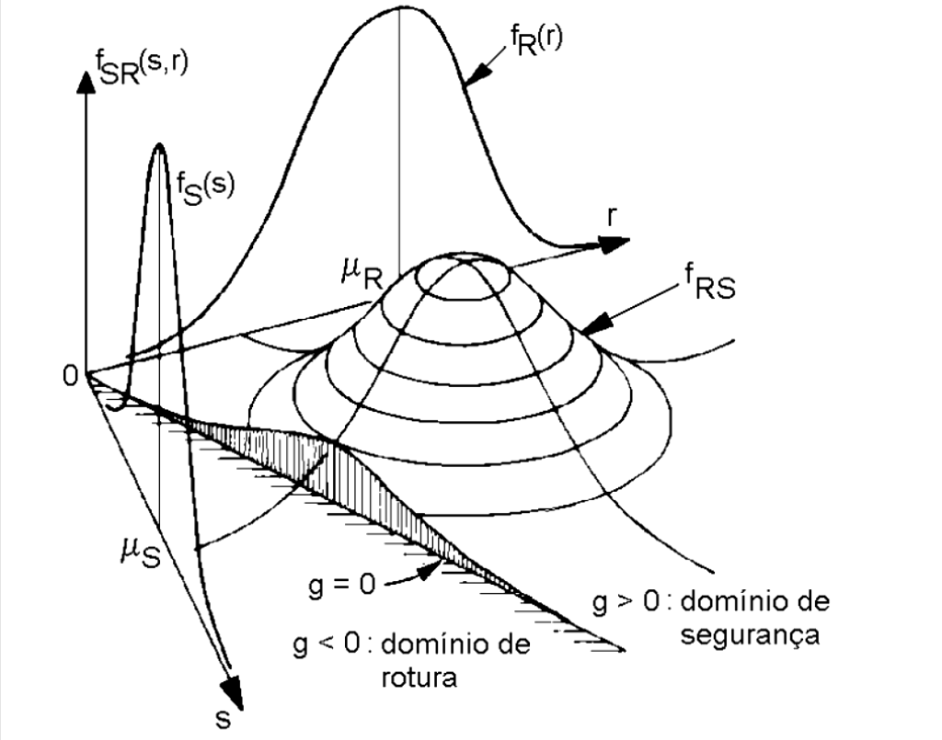


Figura 2. 5 - Representação tridimensional das variáveis envolvidas no problema de fiabilidade estrutural [12]

Sendo $f_X(X)$ a função densidade de probabilidade conjunta do vetor X que inclui a totalidade das variáveis base, a probabilidade de colapso é dada por:

$$p_c = P(g(X) \leq 0) = \int_{g(x)\leq 0} f_x(x)dx = \int_{g(x)\leq 0} F_R(x)f_S(x)dx \tag{2.15}$$

A probabilidade de colapso pode ser determinada através deste integral, designado integral de convolução. O integral representa a totalidade dos casos para os quais a resistência é inferior a uma dada ação, para toda a gama de ações possíveis [12].

O índice de fiabilidade, em função da probabilidade de colapso é dado por:

$$\beta = -\Phi^{-1}(p_c) \tag{2.16}$$

Onde Φ^{-1} é a inversa da função distribuição normal reduzida.

2.1.7. Métodos de Cálculo da Probabilidade de Colapso

A probabilidade de colapso é determinada pela resolução da equação 2.15. Esta resolução apenas é possível quando se conhece a função densidade de probabilidade conjunta.

Existem quatro tipos de métodos para o cálculo da probabilidade de colapso [11]:

1. Integração analítica exata – apenas possível em casos muito simples;
2. Métodos de integração numérica – apenas utilizados em casos onde o número de variáveis é reduzido;
3. Métodos do segundo momento – podem ser do tipo FORM (*First Order Reliability Method*) ou SORM (*Second Order Reliability Method*).
4. Métodos de simulação – o integral definido pela equação 2.15 é determinado por aproximações numéricas.

Os dois primeiros métodos têm uma aplicação muito limitada dada a complexidade dos problemas de análise estrutural [11]. Desta forma os mais utilizados são os métodos do segundo momento e os métodos de simulação, explicando-se de seguida cada um deles.

Neste trabalho são utilizados os métodos de simulação, recorrendo-se ao método de Monte Carlo.

2.1.8. Métodos de Fiabilidade do Segundo Momento

Estes métodos permitem determinar o índice de fiabilidade, por forma a quantificar a segurança de uma estrutura.

Admitindo que as variáveis aleatórias são independentes e com distribuição normal, com médias μ_R e μ_S e desvios padrão σ_R e σ_S, respetivamente para as resistências e para as ações, pode-se determinar o valor médio (μ_g) e o desvio padrão (σ_g) da função de estado limite ($g(X)$).

$$\mu_g = \mu_R - \mu_S \tag{2.17}$$

$$\sigma_g = \sqrt{\sigma_R^2 + \sigma_S^2} \tag{2.18}$$

A probabilidade de colapso é então dada por:

$$p_c = \Phi(-\beta) \tag{2.19}$$

Onde Φ é a função distribuição normal reduzida e β representa o índice de fiabilidade dado por:

$$\beta = \frac{\mu_g}{\sigma_g} \tag{2.20}$$

A equação 2.19 representa o valor exato da probabilidade de colapso quando as variáveis que representam as resistências e as ações têm uma distribuição normal e independente. No entanto, devido à complexidade da maioria das estruturas analisadas em engenharia, as variáveis por norma não são independentes entre si, nem definidas por distribuições normais. Situações de funções de estado limite não lineares, são também habituais. Nestes casos o método anteriormente apresentado não pode ser utilizado. Por forma a resolver esta limitação, surgiram os métodos de fiabilidade de primeira e segunda ordem (FORM e SORM, respetivamente), que permitem resolver problemas mais complexos [11].

O FORM baseia-se numa formulação que sugere a transformação da função de estado limite não linear ($g(u) = 0$) numa função linear ($g'(u) = 0$), em torno do ponto de dimensionamento (u^*) num espaço normalizado [11]. Este método encontra-se ilustrado na figura 2.6.

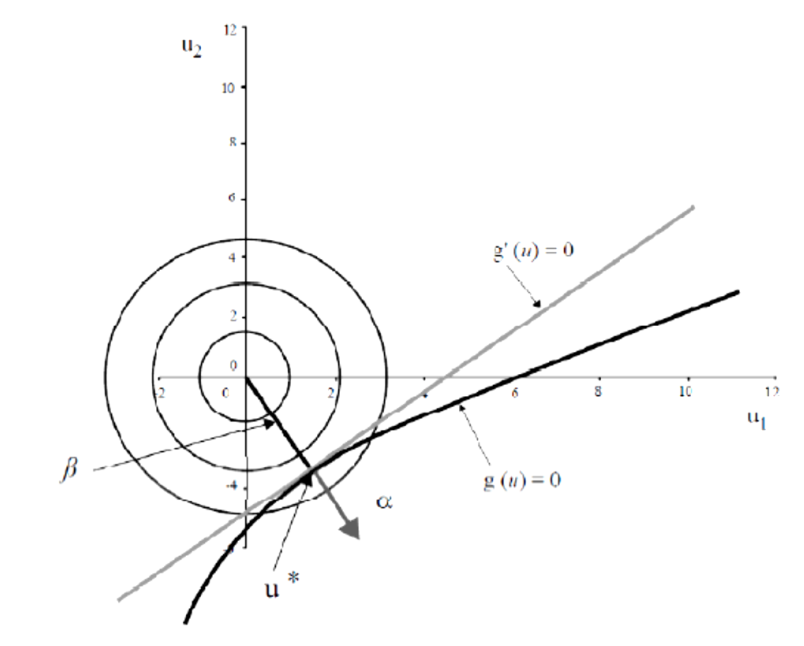


Figura 2. 6 – Ilustração bidimensional do procedimento de aproximação FORM [11]

O índice de fiabilidade é dado pela menor distância entre a origem do sistema de eixos das variáveis aleatórias normalizadas e a função estado limite normalizada ($g'(u)$). α representa um vetor perpendicular à função de estado limite no ponto de dimensionamento.

Dada a não linearidade da função de estado limite, a determinação do ponto de dimensionamento é dada por:

$$\beta = \min_{u \in g(u)=0} \sqrt{\sum_{i=1}^{n} u_i^2} \tag{2.21}$$

Se a função de estado limite for diferenciável, o problema pode ser resolvido através de:

$$\alpha_i = \frac{-\frac{\partial_g}{\partial_{u_i}}(\beta \cdot \alpha)}{\left[\sum_{i=1}^{n}\left(\frac{\partial_g}{\partial_{u_i}}(\beta \cdot \alpha)\right)^2\right]^{\frac{1}{2}}}, i = 1,2,\cdots,n \tag{2.22}$$

$$g(\beta \cdot \alpha_1, \beta \cdot \alpha_2, \cdots, \beta \cdot \alpha_n) = 0 \tag{2.23}$$

Por forma a iniciar o processo iterativo, começa por se arbitrar um ponto de dimensionamento ($u^*=\beta\cdot\alpha$), inserindo o seu valor na equação 2.22, por forma a calcular α, que por sua vez é inserido na equação 2.23, obtendo-se o novo valor β, o que corresponde à primeira iteração. O processo é repetido até se verificar a convergência do índice de fiabilidade β [11].

No entanto, existem problemas em que a aproximação da função de estado limite a uma superfície plana pode conduzir a resultados insatisfatórios, dado algumas destas funções terem curvaturas significativas, ou quando a função de estado limite linear representada por variáveis aleatórias com distribuições originais seja aproximada a uma superfície não linear, aquando da transformação das variáveis aleatórias para um espaço normal reduzido. Nestes casos utiliza-se o método SORM, que recorre a aproximações não lineares da função de estado limite, obtendo assim melhores resultados. Este método utiliza normalmente superfícies paraboloides e esféricas nas aproximações das superfícies de estado limite [11].

2.1.9. Método de Simulação de Monte Carlo

A probabilidade de colapso é definida através do integral da convolução, expresso pela equação 2.15. A determinação deste integral é em muitos casos impossível. Por forma a resolver este problema utilizam-se técnicas de simulação que permitem obter estimativas do integral em problemas para os quais a função de estado limite pode ter qualquer forma e as variáveis aleatórias qualquer distribuição [13].

De uma forma genérica, o método de Monte Carlo simula todas as variáveis aleatórias intervenientes no problema em análise, tendo em conta as respetivas distribuições, e quantifica todas as respostas estruturais associadas aos conjuntos das variáveis simuladas [13].

A utilização do método de Monte Carlo na estimativa da probabilidade de colapso de um sistema consiste em:

1. Gerar uma amostra de variáveis aleatórias definidas por parâmetros estatísticos e por distribuições matemáticas;
2. Calcular a função de estado limite normalmente utilizando um modelo computacional;
3. Verificar a violação da função de estado limite;
4. Repetir os passos 1, 2 e 3, N vezes, contando as ocorrências de colapso, ou seja, o número de vezes que a função de estado limite é violada;
5. Estimar a probabilidade de colapso pela expressão 2.24.

$$p_c = \frac{n_c}{N}$$

(2.24)

Onde n_c representa o número de simulações em que ocorre colapso, e N o número total de simulações.

O número de simulações necessárias para obter resultados fiáveis depende, essencialmente, da probabilidade de colapso e da estrutura em análise, ou seja, da função de estado limite [13]. Desta forma, o número de simulações aumenta significativamente para probabilidades muito reduzidas, aumentando desta forma o esforço computacional, sendo este o principal inconveniente do método de simulação de Monte Carlo.

Torna-se então muito importante definir qual o número de simulações indispensáveis numa avaliação de fiabilidade estrutural. Existem vários limites definidos por autores, tais como, por exemplo, Faber em 2007, que propõe que, para situações em que se pretenda estimar uma probabilidade de colapso na ordem de 10^{-6}, sejam necessárias 10^8 simulações, com coeficiente de variação de 10% [11, 13]. Outros autores dizem ainda que o número de simulações deve estar entre $1/p_c$ e $10/p_c$ [11]. Bronding em 1964 propôs que o número de simulações fosse estimado por:

$$N \geq -\frac{ln(1-c)}{p_c}$$

(2.25)

Onde c traduz o nível de confiança da estimativa da probabilidade de colapso.

Existe também a expressão sugerida por Shooman em 1968, que indica que o erro da probabilidade de colapso, para um número total de simulações, com um nível de confiança de 95% é dado por:

$$erro(\%) = 200 \times \sqrt{\frac{1-p_c}{N \times p_c}}$$

(2.26)

Em suma, a aplicação do método de simulação de Monte Carlo apresenta as seguintes vantagens e desvantagens:

Vantagens

- Este método pode ser aplicado a qualquer estrutura, para todas as distribuições probabilísticas das variáveis aleatórias e para qualquer que seja a configuração da função de estado limite;
- O erro associado a este método pode ser perfeitamente controlado a partir do aumento do número de simulações.

Desvantagens

- Requer um modelo estocástico de todas as variáveis aleatórias envolvidas;
- Devido ao elevado número de simulações, a computação deste método torna-se morosa. Esta desvantagem é particularmente grave quando é necessário executar uma análise de elementos finitos sempre que se calcula $g(x)$. Técnicas alternativas de redução cuidada da variância das variáveis aleatórias, podem tornar o processo mais eficiente.

A associação do método de Monte Carlo às redes neuronais artificiais elimina a última desvantagem apresentada, possibilitando a poupança de muito tempo de computação [7].

O método de Monte Carlo será aplicado nos exemplos estruturais em estudo, por forma a calcular a probabilidade de colapso.

2.2. Redes Neuronais Artificiais

Redes Neuronais Artificiais (RNA) são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neuronal de organismos inteligentes, e que adquirem conhecimento através da experiência. Sendo que "conhecimento" neste contexto se refere à capacidade de relacionar informações de forma coerente. [15]

Nesta secção apresentam-se aspetos gerais sobre estes mecanismos, por forma a clarificar as suas origens, utilidade e modo de aplicação.

2.2.1. Neurónio Biológico

Uma descrição do modelo biológico que serve de base às RNA, permite a compreensão dos princípios de funcionamento e tipos de estruturas das RNA [16].

O neurónio biológico é o elemento básico do sistema nervoso [16].

O cérebro humano é composto por um elevado número de neurónios ligados entre si, que comunicam através de sinais. Existem vários tipos de neurónios com diferentes funções e utilidades no sistema nervoso, possuindo por isso diferentes tamanhos e formas. As formas mais comuns são do tipo piramidal ou esférico, sendo conhecidos 10000 diferentes tipos de neurónios. Na figura 2.7 representa-se um neurónio do tipo piramidal [8].

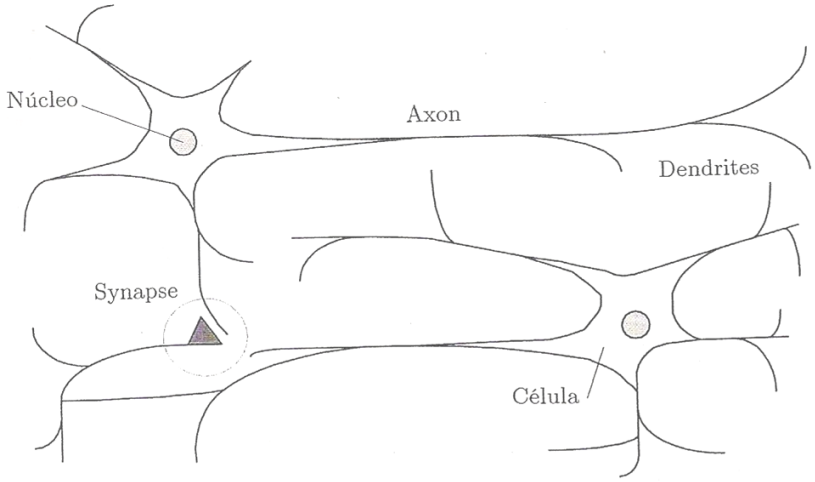


Figura 2.7 - Neurónio biológico [8]

Nos neurónios a informação (sinal elétrico) é transmitida da seguinte forma: as dendrites intersectam os sinais provenientes dos outros neurónios, funcionando como entradas do neurónio. No corpo central existe uma transformação (ativação) do sinal, ou seja, quanto o potencial elétrico no limite da membrana do núcleo atinge um determinado valor, é gerado um potencial de ação que é transmito pelo axon. O axon transmite o sinal recebido do corpo central pelas dendrites de outros neurónios e funciona também como modulador e amplificador de sinal. As sinapses são as ligações existentes entre o axon do neurónio anterior e uma dendrite do neurónio posterior e são elas as responsáveis pela capacidade de aprendizagem [16, 8].

É a interação entre vários neurónios combinados em rede que concede ao cérebro a capacidade de aprender, reconhecer e generalizar [16].

De referir que cada neurónio possui múltiplas entradas, dependendo do número de dendrites, mas apenas possui uma saída [8].

O cérebro humano possui entre 10^{10} e 10^{12} neurónios, existindo aproximadamente 10^{18} sinapses de ligação nas células que o compõem [8].

2.2.2. Neurónio Artificial

O neurónio artificial pretende simular as características do neurónio biológico. Na figura 2.8 representa-se a estrutura básica do neurónio artificial, fazendo-se a analogia deste com o neurónio biológico [8].

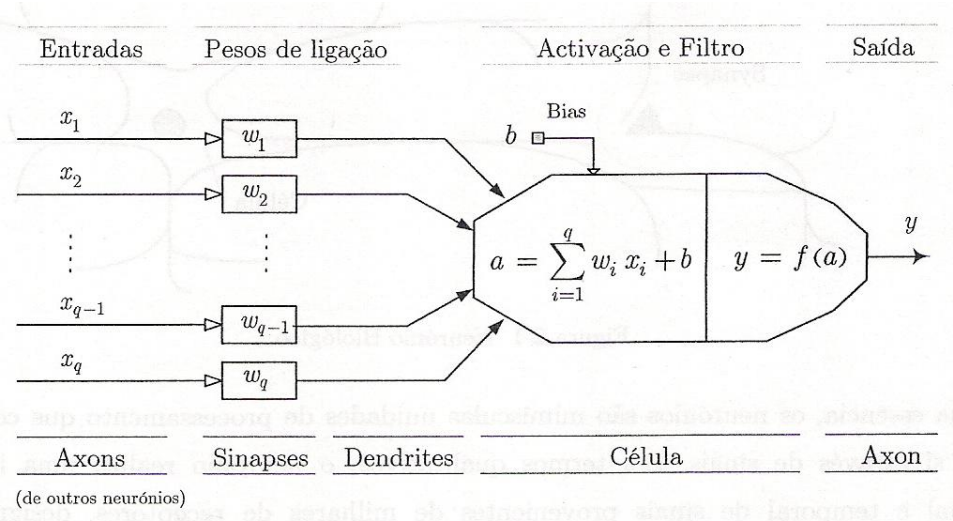


Figura 2.8 – Modelo artificial de um neurónio [8]

Os neurónios artificiais são simples unidades de processamento que podem receber várias entradas de outros neurónios, através das dendrites, produzindo um único sinal de saída que se propaga pelo axon para o neurónio seguinte [16, 8].

A descrição de um neurónio artificial pode ser feita por uma equação matemática muito simples que calcula o valor de saída em função dos valores de entrada. Associado a cada entrada x_i, tem-se um valor w_i, designado por peso de ligação. À soma ponderada das entradas, *a*, dá-se o nome de ativação do neurónio, que depois de transformada por uma função de ativação ou de transferência, *f(a)*, possibilita a obtenção da saída *y*. O bias, *b*, permite obter uma saída não nula quando todas as entradas x_i forem iguais a zero [8].

Os pesos de ligação existentes entre os vários neurónios são parâmetros ajustáveis, que conferem às estruturas neuronais a capacidade de aprendizagem e de colocação da informação na memória. O peso associado a uma determinada ligação entre dois neurónios é multiplicado pela saída do neurónio anterior originando uma das entradas do neurónio posterior, determinando o grau de interação entre dois neurónios artificiais [16].

Uma ligação pode funcionar como excitação ou inibição para um neurónio consoante os pesos de ligação forem positivos ou negativos, respetivamente [16, 8].

Distinguem-se duas fases para o neurónio artificial: a fase de treino, no qual o neurónio aprende que resposta fornecer consoante o sinal de entrada, e a fase de operação ou simulação, no qual o neurónio responde de acordo com que foi preparado na fase de treino [8].

2.2.3. Funções de Ativação

Diferentes funções de ativação produzem diferentes saídas. Assim sendo, dependendo da aplicação em causa, é escolhida a função de ativação [8]. As funções de ativação determinam o nível de transformação de um neurónio perante as suas entradas [16]. É desejável que estas funções se comportem como um filtro, ou seja, devem ser limitadas superior ou inferiormente para que os valores calculados à saída dos neurónios não apresentem comportamentos instáveis [16,8].

As funções de transferência mais utlizadas são: funções lógicas, funções lineares e semi-lineares, funções sigmoidais e funções gaussianas [8].

Funções lógicas

Normalmente estão-lhe associadas as saídas (+1) e (0) sendo neste caso também designadas funções binárias, ou (+1) e (-1) sendo conhecida por função bipolar lógica. Expressam-se da seguinte forma [8]:

$$f(x) = \begin{cases} +1 & , \mathrm{x} \geq 0 \\ 0 \text{ ou } (-1) & , \mathrm{x} < 0 \end{cases} \tag{2.27}$$

Embora fáceis de implementar, são limitadas a nível de aprendizagem e desempenho.

Funções lineares e semi-lineares

Para este tipo de funções, existem limites superior e inferior à função, excetuando a função linear simples cuja saída é função linear da entrada. O tipo mais comum é a função linear com saturação, descrita por [8]:

$$f(x) = \begin{cases} 0 & , \mathrm{x} \leq 0 \\ \mathrm{x} & , 0 \leq \mathrm{x} \leq 1 \\ 1 & , \mathrm{x} \geq 1 \end{cases} \tag{2.28}$$

Estas funções são não diferenciáveis nos pontos de transição.

Funções sigmoidais

Este tipo de funções são contínuas e diferenciáveis em todo o seu domínio, destacando-se a função sigmoidal unipolar, ilustrada na figura 2.9 e expressa por [8]:

$$f(x) = \frac{1}{1 + e^{-\alpha x}} \, , \alpha \in \Re \tag{2.29}$$

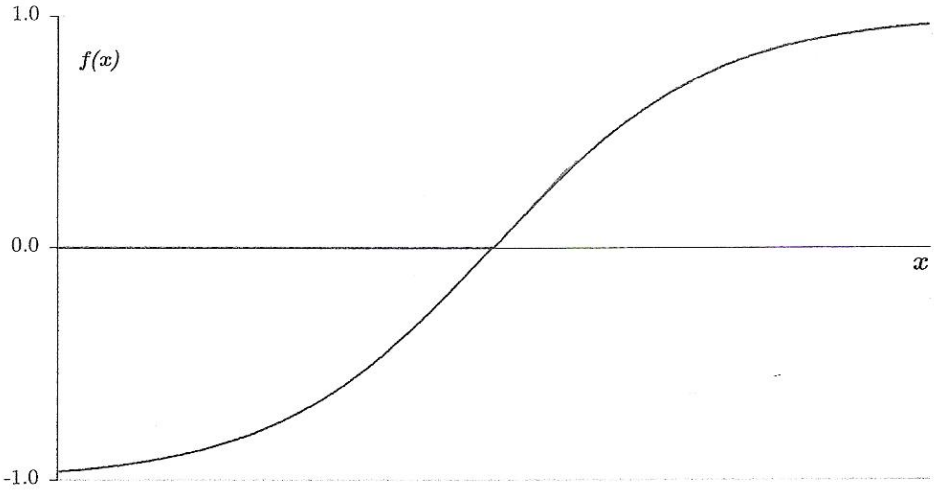


Figura 2. 9 – Representação gráfica da função sigmoidal. Adaptado de [8]

Estas funções são limitadas superior e inferiormente, com valores aproximadamente constante para entradas acima ou abaixo desses limites. Esta função possui uma região de transição cuja extensão pode ser alterada variando o valor da constante α.

Outras funções com as mesmas propriedades são igualmente implementadas nas estruturas neuronais artificiais, como é o caso da função tangente hiperbólica.

Função radial

Mais utilizadas em problemas de controlo, as funções radiais são também designadas funções gaussianas. Este tipo de função tem um valor máximo para o centro de transferência e zero para valores de entrada desfasados desse centro, sendo estes valores ajustáveis de modo a obter o comportamento pretendido [16, 8].

2.2.4. Revisão Histórica

O primeiro modelo artificial de um neurónio biológico surgiu em 1943, em artigos de Warren McCulloch, neuroanatomista e psiquiatra, e Walter Pitts, matemático [17]. Apresentaram um modelo matemático no qual definiram um neurónio artificial em analogia com um neurónio biológico.

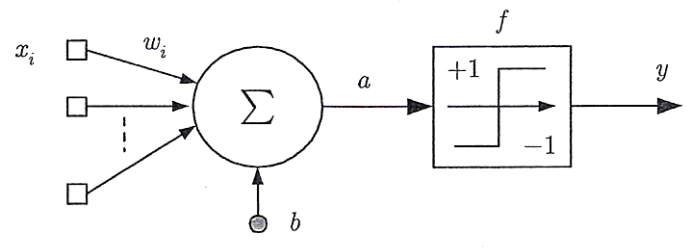


Figura 2.10 - Neurónio de McCulloch e Pitts [8]

Como representado na figura 2.10, foi utilizada a função binária.

Em 1949 Donald Hebb, através do seu livro "The Organization of Behavior", introduziu o conceito de aprendizagem. O princípio "quando a estrada sináptica e a saída de um neurónio estão ambas ativas, ocorre uma mudança física caracterizada pelo aumento na força dessa ligação", também conhecido por Regra de Hebb, foi base de vários algoritmos de treino.

Em 1958, Frank Rosenblatt [18] criou o perceptrão, que consistia num modelo baseado na analogia com a retina do olho. Rosenblatt mostrou a capacidade do perceptrão em classificar padrões linearmente separáveis.

No final da década de 60, Windrow e Hoff [19] apresentaram o modelo ADELINE (ADAptive LINear Element), capaz de atuar como filtro de sinais, ordenando padrões em duas categorias. Estes autores introduziram regra do delta, designada *least mean square rule (LMS)*, como regra de aprendizagem baseada no método dos mínimos quadrados.

Em 1969, Minsky e Papert [20] identificaram as limitações dos métodos existentes na altura, mostrando que apenas os problemas linearmente separáveis podiam ser classificados.

Dadas as limitações identificadas e as inerentes aos computadores da altura, nos anos 70 não ocorreram progressos nesta área.

Nos anos 80, Hopfield [21], Prémio Nobel da Física, demonstrou que uma rede opera no sentido de minimizar a energia que lhe está associada.

Rumelhart *et al.* [22] propuseram o algoritmo de retropropagação (*backpropagation*) para treinar redes multicamada (*multilayer neural networks*).

As redes neuronais têm vindo a ser um tema em evolução, fazendo com que a sua aplicação se generalize em várias áreas, como a biologia, investigação policial, medicina, economia e finanças, engenharia, destacando-se a sua aplicação em processamento de sinais, reconhecimento de padrões complexos e tarefas de classificação.

2.2.5. Tipos de Redes Neuronais Artificiais

A estrutura de uma rede neuronal artificial determina as funções para as quais a rede melhor se aplica [16, 8]. As redes são classificadas de acordo com a sua topologia, ou seja de acordo com o número de neurónios, as ligações entre neurónios e as funções de ativação.

As camadas e a conectividade, são os principais aspetos que caracterizam o tipo de rede, conforme se mostra na tabela 2.5:

Tabela 2.5 – Caracterização dos tipos de rede neuronal [8]

Topologia de rede	**Camadas**	**Número de camadas**	**Camada simples** **Múltiplas camadas**
		Número de neurónios	**Um neurónio** **Vários neurónios**
	Conectividade	**Grau de conectividade**	**Ligação parcial** **Ligação total**
		Sentido de conectividade	**Para a frente** **Para trás** **Lateral** **Auto ligação**
		Tipo de conectividade	**Excitatória** **Inibitória**

As arquiteturas de rede mais importantes são descritas de seguida.

Redes de camada única

Neste tipo de rede, os neurónios encontram-se todos no mesmo nível, servindo como entrada e saída da rede. É possível a existência de neurónios da mesma camada ligados entre si, tratando-se neste caso das chamadas conexões laterais.

Redes de camadas unidirecionais

Neste tipo de redes, também chamadas de redes diretas ou de avanço simples, os neurónios estão organizados por camadas e as ligações dão-se entre duas camadas sucessivas e sempre da camada anterior para a posterior.

Redes recorrentes

As redes recorrentes, podem ter uma ou várias camadas, e as ligações podem ser feitas entre neurónios de camadas anteriores, da própria camada ou de camadas posteriores, sendo este tipo de redes muito utilizado em problemas de otimização.

Redes estáticas

A estrutura definida à partida não é alterada no decorrer do processo de treino da rede. Apenas os pesos das ligações sofrem alterações.

Redes dinâmicas

Neste tipo de redes ocorrem alterações a nível do número de neurónios e de ligações. Estas alterações são feitas de acordo com critérios previamente determinados.

2.2.6. Treino de Redes Neuronais Artificiais

O processo de treino ou aprendizagem de uma rede neuronal artificial, é o processo de determinação dos parâmetros internos da rede, por forma a que esta apresente o comportamento desejado [8].

Os métodos de treino da rede neuronal podem ser divididos em três categorias: treino não supervisionado, treino por reforço crítico e treino supervisionado [8].

Treino não supervisionado

Neste tipo de treino o ajuste dos pesos da rede é feito de modo a que a rede responda de modo semelhante quando sujeita a estímulos semelhantes [23]. Ou seja, sempre que se apresenta uma nova entrada à rede neuronal artificial, existe uma competição entre os neurónios da camada de saída, através do ajuste dos pesos de ligação para representar a entrada apresentada naquele momento. Considera-se que a rede se encontra bem treinada quando esta tem a capacidade de reconhecer todas as entradas apresentadas durante a fase de treino [8].

Treino por reforço crítico

Neste tipo de treino utiliza-se apenas um conjunto de valores de entrada no processo de aprendizagem da rede, não sendo necessários os valores das saídas correspondentes a esses valores de entrada. Um valor obtido à saída da rede (sinal de reforço), permite que a rede favoreça os comportamentos corretos e penalize os incorretos, acertando desta forma os valores dos pesos da rede [8].

O processo de treino é feito da seguinte forma: inicialmente atribuem-se valores aleatórios aos pesos e obtêm-se os valores de saída. De seguida, avalia-se a saída e o sinal de reforço é inserido na rede, aumentando o valor dos pesos que favoreceram o desempenho e diminuindo os outros [8].

Treino supervisionado

Sendo o tipo de treino mais utilizado nos diferentes tipos de redes neuronais existentes, este tipo de treino utiliza um conjunto de dados de entrada e respetivas saídas e aplica regras baseadas na minimização do erro, permitindo calcular os valores dos pesos de ligação a partir de informação disponível do sistema, comparando os valores à saída da rede neuronal, com os valores esperados. Este processo tem em vista encontrar o melhor conjunto de valores dos pesos de ligação que minimize o erro entre os valores conhecidos e os valores previstos pela rede.

Exemplos importantes deste tipo de treino são a regra de aprendizagem do perceptrão, a regra do delta e o algoritmo de retropropagação. Pela sua importância nesta dissertação, apresenta-se com mais detalhe este último algoritmo aplicado ao treino de redes multicamada de avanço simples (*Multilayer Feedforward Network*).

Este tipo de rede neuronal é usada em muitas aplicações, tanto para reconhecimento de padrões como para aproximar funções. Com efeito, vários autores, Cybenko [24], Funahashi [25],Hornik et al. [26] e Hornik [27], provaram ser possível aproximar, com grau de precisão arbitrário, qualquer função contínua não linear, utilizando o treino adequado e um número de neurónios suficientes na camada interna e com função de ativação do tipo sigmoidal [8].

Considera-se uma rede com *M* camadas, sendo a entrada da rede, a primeira camada (*m*=1), que apenas transporta a informação de entrada de cada neurónio para a camada seguinte, não tendo qualquer função de transformação, e a saída, a última camada (*m*=*M*), sendo as restantes (*M* – 2) camadas, designadas por camadas internas. Considera-se ainda que N_m representa o número de neurónios da camada *m* e que x_i^m representa a saída do neurónio *i* da camada *m*.

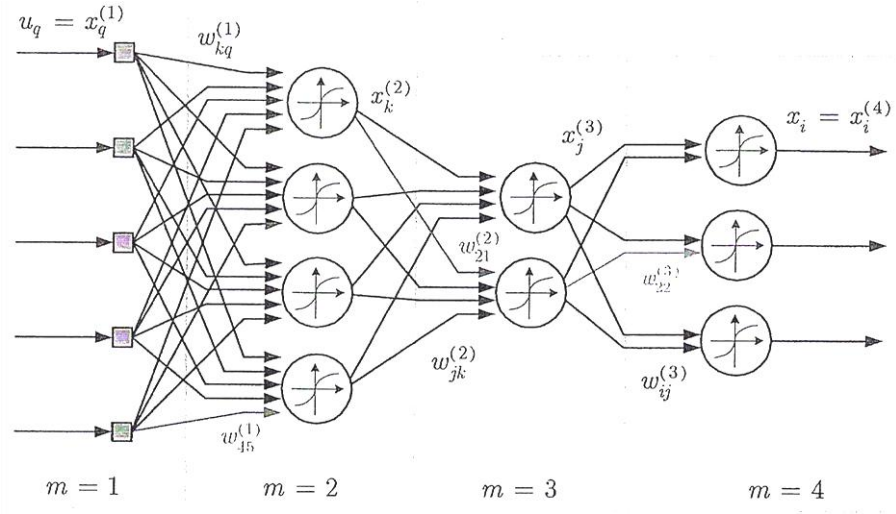


Figura 2.11 – Rede neuronal artificial com 4 camadas [8]

Tomando como exemplo a rede apresentada na figura 2.11, tem-se $M = 4$, $N_1 = 5$, $N_2 = 4$, $N_3 = 2$ e $N_4 = 3$.

Utilizando as definições apresentadas em 2.2.2., obtém-se as seguintes expressões para o caso geral das camadas $m = 2,\ldots,M$.

$$a_i^{(m)} = \sum_{j=1}^{N_{m-1}} w_{ij}^{(m-1)} x_j^{(m-1)} + b_i^{(m)} = \sum_{j=0}^{N_{m-1}} w_{ij}^{(m-1)} x_j^{(m-1)} \tag{2.30}$$

Em que se considera $w_{i0}^{(m-1)} = b_i^{(m)}$ e $x_0^{(m-1)} = 1$.

A expressão 2.31 representa a saída do neurónio *i* da camada *m*, na qual se considera que todos os neurónios têm a mesma função de ativação.

$$x_i^{(m)} = f_i^{(m)}\left(a_i^{(m)}\right) = f_m\left(a_i^{(m)}\right) \tag{2.31}$$

Algoritmo de retropropagação (*Backpropagation*)

Este algoritmo, considerado uma generalização da regra do delta, surgiu como forma de resolver os problemas inerentes ao ajuste dos pesos para os neurónios das camadas internas [8].

Durante o treino da rede o algoritmo funciona da seguinte forma: numa primeira fase o sinal à entrada de cada neurónio da primeira camada é propagado através da rede, obtendo-se a resposta produzida pela camada de saída. Numa segunda fase, a resposta obtida à saída da rede é comparada à saída desejada, calculando-se o erro. Na terceira fase, este erro é retropropagado através da rede e os pesos são ajustados, por forma a produzir a resposta correta [28].

Para uma rede neuronal multicamada de avanço simples, como a da figura 2.11, com *M* camadas e N_m neurónios, o erro da resposta do neurónio *i* da camada de saída para o padrão de treino *k*, define-se pela diferença entre x_i^d (saída desejada) e $x_i = x_i^{(M)}$ (saída real).

$$e_i^k = \left(x_i^d - x_i^{(M)}\right)_k = \left(x_i^d - x_i\right)_k \tag{2.32}$$

A função de erro que se pretende minimizar é a função soma do erro quadrático. Para tal, calcula-se o erro para o *k*-ésimo conjunto de dados de entrada e saída do conjunto de treino:

$$E_k = \sum_{i=1}^{N_M}\left[\frac{1}{2}\left(e_i^k\right)^2\right]_k \tag{2.33}$$

Calculando-se de seguida, a função que se pretende minimizar:

$$E = \sum_{k=1}^{P} E_k = \sum_{k=1}^{P}\sum_{i=1}^{N_M}\left[\frac{1}{2}\left(x_i^d - x_i\right)^2\right]_k \tag{2.34}$$

A expressão 2.34 representa a soma do erro quadrático médio para *P* conjuntos de treino.

Nos cálculos seguintes considera-se a minimização para cada conjunto *P*.

O ajuste dos pesos das camadas *m* e (*m*+1) é definido de modo proporcional ao negativo da derivada do erro em relação ao peso. Sendo $\Delta w_{ij}^{(m)}$ a variação feita no peso $w_{ij}^{(m)}$ da ligação entre o neurónio j da camada m e o neurónio i da camada $(m+1)$ e η uma constante denominada coeficiente de aprendizagem, tem-se:

$$\Delta w_{ij}^{(m)} = -\eta \frac{\partial E}{\partial w_{ij}^{(m)}}, \qquad \eta > 0 \tag{2.35}$$

Derivando a função do erro em relação ao peso, tem-se:

$$\frac{\partial E}{\partial w_{ij}^{(m)}} = \frac{\partial E}{\partial x_i^{(m+1)}} \frac{\partial x_i^{(m+1)}}{\partial w_{ij}^{(m)}} \tag{2.36}$$

Da expressão 2.31, derivando parcialmente a saída do neurónio *i* da camada (*m*+1) em relação ao peso, utilizando a regra da função composta tem-se:

$$\frac{\partial x_i^{(m+1)}}{\partial w_{ij}^{(m)}} = \frac{\partial f_{m+1}\left(a_i^{(m+1)}\right)}{\partial w_{ij}^{(m)}} = \frac{\partial f_{m+1}\left(a_i^{(m+1)}\right)}{\partial a_i^{(m+1)}} \frac{\partial a_i^{(m+1)}}{\partial w_{ij}^{(m)}} = f'_{m+1} \frac{\partial a_i^{(m+1)}}{\partial w_{ij}^{(m)}} \tag{2.37}$$

Considerando 2.30, fica-se com:

$$\frac{\partial x_i^{(m+1)}}{\partial w_{ij}^{(m)}} = f'_{m+1} \frac{\partial a_i^{(m+1)}}{\partial w_{ij}^{(m)}} = f'_{m+1} \frac{\partial}{\partial w_{ij}^{(m)}} \left(\sum_{j=0}^{N_m} w_{ij}^{(m)} x_j^{(m)} \right) = f'_{m+1}\, x_j^{(m)} \tag{2.38}$$

Em relação ao outro termo da derivada composta da equação 2.36, considera-se que se pretende ajustar os pesos entre a camada ($M-1$) e a camada *M* (camada de saída). Desta forma, utilizando a definição de erro:

$$\frac{\partial E}{\partial x_i^{(M)}} = \frac{\partial}{\partial w_{ij}^{(m)}}\left[\frac{1}{2}\sum_{i=1}^{N_m}\left(x_i^d - x_i^{(M)}\right)^2\right] = -\left(x_i^d - x_i^{(M)}\right)$$

(2.39)

Substituindo (2.38) e (2.39) na equação 2.35, obtém-se:

$$\Delta w_{ij}^{(M-1)} = -\eta\, \alpha_i^{(M)}\, x_j^{(M-1)}$$

(2.40)

Nesta regra de ajuste dos pesos, semelhante à regra do delta, $\alpha_i^{(M)}$ é designado por sinal de erro generalizado, expresso por:

$$\alpha_i^{(M)} = -\left(x_i^d - x_i^{(M)}\right) f'_M$$

(2.41)

Por forma a obter o ajuste dos pesos entre a camada ($M - 2$) e a camada ($M - 1$) e partindo da definição de erro, obtém-se por derivação composta:

$$\frac{\partial E}{\partial x_i^{(M-1)}} = \sum_{n=1}^{N_M} \frac{\partial E}{\partial x_n^{(M)}} \frac{\partial x_n^{(M)}}{\partial x_i^{(M-1)}}$$

(2.42)

Considerando (2.39) e (2.30), para o primeiro e segundo termos do somatório respetivamente, obtém-se:

$$\frac{\partial x_n^{(M)}}{\partial x_i^{(M)}} = \frac{\partial f_M\left(a_n^{(M)}\right)}{\partial a_n^{(M)}} \frac{\partial a_n^{(M)}}{\partial x_i^{(M-1)}} = f'_M \frac{\partial}{\partial x_i^{(M)}}\left(\sum_{j=0}^{N_{M-1}} w_{nj}^{(M-1)} x_j^{(M-1)}\right) = f'_M\, w_{ni}^{(M-1)}$$

(2.43)

Substituindo (2.39) e (2.43) em (2.42), fica-se com:

$$\frac{\partial E}{\partial x_i^{(M-1)}} = \sum_{n=1}^{N_M}\left[-\left(x_i^d - x_i^{(M)}\right) f'_M\, w_{ni}^{(M-1)}\right] = \sum_{n=1}^{N_M}\left[\alpha_n^{(M)} w_{ni}^{(M-1)}\right]$$

(2.44)

Considerando a expressão desta derivada na equação (2.36), tem-se:

$$\frac{\partial E}{\partial x_i^{(M-2)}} = \frac{\partial E}{\partial x_i^{(M-1)}} \frac{\partial x_i^{(M-1)}}{\partial w_{ij}^{(M-2)}} = f'_{M-1}\ x_j^{(M-2)} \sum_{n=1}^{N_M} \left[\alpha_n^{(M)} w_{ni}^{(M-1)}\right]$$

(2.45)

Substituindo (2.45) em (2.35), obtém-se a regra de ajuste dos pesos entre as camadas (*M* – 2) e (*M* – 1), dado por:

$$\Delta w_{ij}^{(M-2)} = -\eta\ \alpha_i^{(M-1)}\ x_j^{(M-2)}$$

(2.46)

Em que $\alpha_i^{(M-1)}$ se obtém por:

$$\alpha_i^{(M-1)} = f'_{M-1} \sum_{n=1}^{N_M} \left[\alpha_n^{(M)} w_{ni}^{(M-1)}\right]$$

(2.47)

Em redes com várias camadas internas, para cada camada *m* tem de se calcular $\alpha_i^{(m)}$. Constatou-se não ser necessário usar mais do que duas camadas intermédias para obter redes neuronais eficientes. A generalidade das redes multicamada utilizadas atualmente tem uma ou duas camadas intermédias. No caso do MATLAB apenas é permitida uma camada intermédia.

Generalizando as equações 2.44 e 2.45,

$$\Delta w_{ij}^{(m)} = -\eta\ \alpha_i^{(m+1)}\ x_j^{(m)}$$

(2.48)

Em que $\alpha_i^{(m+1)}$ se obtém por:

$$\alpha_i^{(m+1)} = f'_{m+1} \sum_{n=1}^{N_{m+2}} \left[\alpha_n^{(m+2)} w_{ni}^{(m+1)}\right]$$

(2.49)

Supondo a utilização da função sigmoidal unipolar em todas as camadas ativas, tem-se:

$$x_i^{(m)} = f_m\left(a_i^{(m)}\right) = \frac{1}{1 + e^{-a_i^{(m)}}} \tag{2.50}$$

Derivando:

$$f'_m = x_i^{(m)}\left(1 - x_i^{(m)}\right) \tag{2.51}$$

O sinal de erro generalizado, $\alpha_i^{(m+1)}$, para as ligações entre a primeira e última camada interna, é dado por:

$$\alpha_i^{(m+1)} = x_i^{(m+1)}\left(1 - x_i^{(m+1)}\right)\sum_{n=1}^{N_{m+2}}\left[\alpha_n^{(m+2)} w_{ni}^{(m+1)}\right] \tag{2.52}$$

O sinal de erro generalizado, $\alpha_i^{(M)}$, para as ligações entre a última camada interna e a camada de saída, é dado por:

$$\alpha_i^{(M)} = x_i^{(M)}\left(x_i^d - x_i^{(M)}\right)\left(1 - x_i^{(M)}\right) \tag{2.53}$$

Este algoritmo tem como vantagem a sua simplicidade, permitindo que seja implementado rapidamente. A morosidade da sua convergência torna-se numa desvantagem à sua aplicação.

Técnicas heurísticas e técnicas de otimização numérica, são utilizadas por forma a melhorar o desempenho deste algoritmo, aumentado desta forma a velocidade de treino.

Nas técnicas heurísticas, destacam-se o método do impulso e o método do passo variável, ficando aqui apenas a referência as estes métodos, não se desenvolvendo mais sobre o tema ao longo desta dissertação.

Em relação a técnicas de otimização numérica, tem-se o método de otimização de Newton, que se obtém, substituindo o coeficiente η, pela inversa da matriz Hessiana *H*, na expressão 2.35. Tem-se:

$$H \equiv \nabla^2 E(w)$$

(2.54)

Em que *E(w)* é a função que se pretende minimizar.

Na iteração (*n*+1) tem-se:

$$w(n+1) = w(n) - H^{-1}(n)\nabla E\big(w(n)\big)$$

(2.55)

Pela equação 2.33, sendo *E(w)* a soma de funções quadráticas,

$$E(w) = \frac{1}{2}e_i^2$$

(2.56)

O elemento *j* do gradiente é dado por:

$$[\nabla E(w)]_j = \frac{\partial E(w)}{\partial w_j} = e_i(w)\frac{\partial e_k(w)}{\partial w_j}$$

(2.57)

O gradiente pode ser descrito por:

$$\nabla E(w) = J^T(w)\, e(w)$$

(2.58)

Sendo *J(w)* a matriz Jacobiana.

O elemento *kj* da matriz Hessiana é:

$$H_{kj} = [\nabla^2 E(w)]_{kj} = \frac{\partial^2 E(w)}{\partial w_k \partial w_j} = \frac{\partial e_i(w)}{\partial w_k}\frac{\partial e_i(w)}{\partial w_j} + e_i(w)\frac{\partial^2 e_i(w)}{\partial w_k \partial w_j}$$

(2.59)

Podendo a matriz Hessiana ser escrita na seguinte forma matricial:

$$\nabla^2 E(w) = J^T(w)\, J(w) + S(w)$$

(2.60)

$$S(w) = e_i(w)\, \nabla^2 e_i(w)$$

(2.61)

Embora mais rápido este método tem desvantagens, nomeadamente a necessidade de utilizar informação de segunda ordem e de calcular a inversa da matriz Hessiana.

O método de Gauss-Newton calcula uma aproximação, assumindo *S(w)* desprezável em relação ao primeiro termo, obtendo-se:

$$H = \nabla^2 E(w) \cong J^T(w)\, J(w) \tag{2.62}$$

Ficando na iteração (*n*+1):

$$w(n+1) = w(n) - \left[J^T\big(w(n)\big)\, J\big(w(n)\big)\right]^{-1}\left[J^T\big(w(n)\big)\right]\left[e\big(w(n)\big)\right] \tag{2.63}$$

No entanto, a necessidade de calcular a inversa da matriz Hessiana mantém-se. Uma solução para este problema foi apresentada por Levenberg-Maquardt. O método de Levenberg-Maquardt consiste na adição de uma restrição adicional $\mu I(w)$, onde μ é uma constante real positiva e I uma matriz identidade com dimensões apropriadas à matriz $[J^T(w)\, J(w)]$. Desta forma, para a iteração (*n*+1), o algoritmo é dado por:

$$\Delta w(n) = -\left[J^T\big(w(n)\big)\, J\big(w(n)\big) + \mu(n) I\right]^{-1}\left[J^T\big(w(n)\big)\right]\left[e\big(w(n)\big)\right] \tag{2.64}$$

Com um aumento de $\mu(n)$, o algoritmo aproxima-se do método do gradiente descendente com um coeficiente de aprendizagem pequeno, e com $\mu(n)$ próximo de zero, aproxima-se do método de Gauss-Newton.

Por norma o algoritmo inicia com um valor pequeno de μ (por exemplo 0,01). Se a iteração não conduzir a um valor mais pequeno da função, essa iteração repete-se com o valor de μ multiplicado por ϑ>1. No caso de a iteração conduzir a um valor mais pequeno da função, o valor de μ é dividido por ϑ>1 na iteração seguinte, obtendo-se desta forma uma convergência mais rápida. O método de Levenberg-Maquardt é especialmente apropriado em problemas mal condicionados.

Nos exemplos práticos serão utilizados redes multicamadas (*multilayer*) progressivas (*feedforward*) e treino supervisonado através do algoritmo de Levenberg-Maquardt , justificando desta forma a descrição detalhada deste algoritmo.

2.3. Hipercubo Latino

Para gerar conjuntos de treino e teste da rede neuronal é necessário garantir que o domínio das variáveis é convenientemente coberto.

Em sistemas estruturais complexos, cuja análise exige elevado esforço computacional, o número de amostras tem de ser reduzido, mas deve permitir uma elevada precisão de resultados. Para tal recorre-se a técnicas de redução da correlação estatística, sendo um exemplo destas técnicas, a técnica do hipercubo latino [29].

Esta técnica, apresentada em 1979 por McKay [30], garante que as amostras utilizadas na simulação cobrem todo o domínio do problema, através da divisão das funções de densidade de probabilidade, de cada variável aleatória, em intervalos de igual probabilidade, e no sorteio de valores dentro desse intervalo [30, 31].

Utiliza-se o trabalho de Olson [30], por forma a explicar como funciona esta técnica.

No método do hipercubo latino a precisão desejada da função de distribuição determina o número de realizações necessárias. Seja N o número de realizações e K o número de variáveis aleatórias. Seja $\boldsymbol{P}$ uma matriz, $N \times K$, em que cada uma das colunas K é uma permutação aleatória de 1,…,N. $\boldsymbol{R}$ é uma matriz $N \times K$, de números aleatórios independentes a partir de uma distribuição uniforme. Estas matrizes formam o plano de amostragem base, representado pela matriz $\boldsymbol{S}$ como:

$$S = \frac{1}{N}(P - R) \tag{2.65}$$

Cada elemento de $\boldsymbol{S}$, S_{ij}, é então mapeado de acordo com a distribuição marginal como:

$$\hat{x}_{ij} = F_{xj}^{-1}(S_{ij}) \tag{2.66}$$

Onde F_{xj}^{-1} representa o inverso da função de distribuição cumulativa para a variável de destino j. O vetor $\hat{x}_{ij} = [\hat{x}_{i1}\ \hat{x}_{i2}\ \cdots\ \hat{x}_{ik}]$ contém agora dados de entrada para uma computação determinística.

Um possível plano de amostragem para duas variáveis de entrada e cinco realizações é ilustrado na figura 2.12. Note-se que a amostra está espalhada por todo o espaço amostral, contendo uma imagem de cada linha e de cada coluna. Na técnica do hipercubo latino, o espaço amostral de cada variável é dividido em intervalos, sendo selecionado apenas um valor de cada intervalo, como se representa na figura 2.12.b.

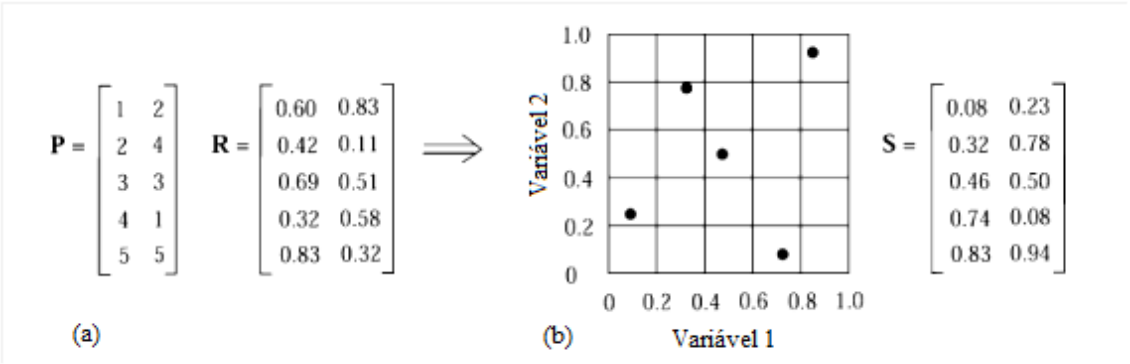


Figura 2. 12 – Hipercubo latino, duas variáveis e cinco realizações. A matriz 5×2 (a) determina o plano ilustrado em (b) [30]

Mesmo que a distribuição marginal de cada variável seja representada de forma eficiente, existe o risco de aparecimento de correlações não desejadas, como representado na figura 2.13 (a).

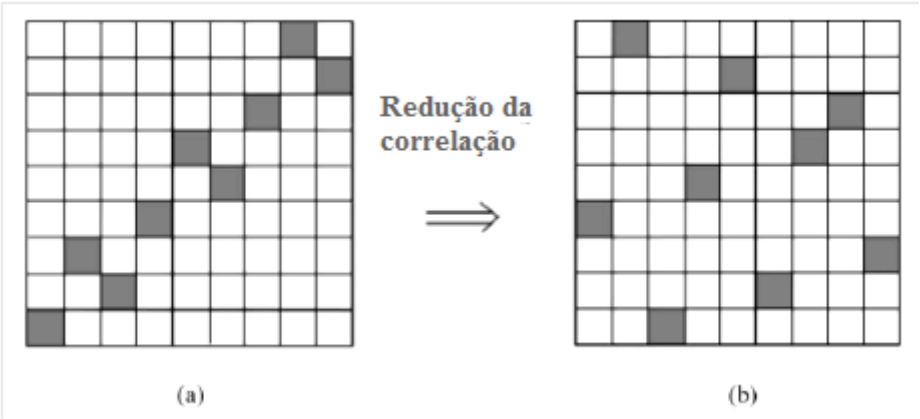


Figura 2. 13 – Amostragem com correlação (a) e sem correlação (b) [30]

Tal correlação poderá ser reduzida pela modificação na permutação da matriz ***P***. Os elementos da matriz ***P***, p_{ij}, são divididos por N+1 e mapeados sobre uma distribuição Gaussiana com média igual a zero e desvio padrão igual a um, da seguinte forma:

$$y_{ij} = \Phi_{(0,1)}^{-1}\left(\frac{p_{ij}}{N+1}\right) \tag{2.67}$$

Em seguida a covariância da matriz ***Y*** é estimada e decomposta em matrizes triangulares usando a decomposição de Cholesty:

$$\overline{LL}^T = cov(Y) \tag{2.68}$$

Onde $\overline{\boldsymbol{L}}$ é uma matriz triangular inferior. Uma nova matriz ***Y**** com covariância de amostra igual à identidade é calculada da seguinte forma:

$$Y^* = Y\left(\overline{L}^{-1}\right)^T \tag{2.69}$$

Os elementos das colunas de ***Y**** são usados como elementos das colunas da matriz ***P****. Se os elementos da matriz ***P*** na equação 2.67 forem substituídos pelos elementos desta matriz, a amostra da matriz ***S*** irá conter uma quantidade consideravelmente menor de correlações indesejadas. Note-se que a decomposição de Cholesky da equação 2.68 exige que *cov(Y)* seja positiva, o que por sua vez exige que o número de realizações seja superior ao número de variáveis aleatórias (*N* > *K*). A figura 2.13 ilustra o efeito da correlação de um plano de amostragem de duas variáveis onde (a) representa o plano de amostragem antes da correlação e (b) o plano de amostragem após a redução de correlação.

Se a matriz de correlação alvo for diferente da unidade, a equação 2.69 é substituída por:

$$Y^* = Y\left(\overline{L}^{-1}\right)^T L^T \tag{2.70}$$

Onde *L* é uma matriz triangular inferior correspondente à decomposição de Cholesky da matriz de correlação destino.

A correlação da amostra da equação 2.66 é exatamente a correlação de destino se as variáveis estocásticas forem gaussianas e será um valor aproximado se as variáveis estocásticas não forem gaussianas. Um algoritmo iterativo poderá ser empregue para melhorar a correlação, no caso de a correlação não ser gaussiana. O processo de redução da correlação pode introduzir algum viés, não resultando num estimador centrado.

Esta técnica é utilizada no exemplo da placa reforçada para gerar os ficheiros necessários ao treino da rede neuronal artificial, onde, devido ao elevado número de variáveis, seria necessário um enorme esforço computacional e com isto elevado gasto de tempo, que são desta forma diminuídos. Para a aplicação desta técnica utiliza-se a função *lhsdesign* do MATLAB que permite implementar a técnica do hipercubo latino e ainda torná-la mais eficiente, através da opção de reduzir a correlação.

2.4. Otimização Estrutural

A palavra "otimização" é definida como "fazer tão perfeito, efetivo ou funcional quanto possível" [32].

A otimização estrutural engloba um conjunto de teorias e métodos que procuram obter a estrutura que desempenha mais eficientemente as funções para as quais é projetada.

Desde os anos 60 que esta área tem vindo a ser estudada, e apresenta resultados satisfatórios na construção civil, indústria automobilística e indústria aeroespacial. Estes projetos têm como objetivo básico a redução do custo financeiro e da quantidade de material utilizado, e, além disso, a garantia de que a estrutura suporte todas as restrições mecânicas impostas, sejam elas estáticas ou dinâmicas [32].

Na década de 70, quando se interligaram algoritmos de programação matemática e programas de elementos finitos, a otimização estrutural desenvolveu-se bastante. Atualmente são muito utilizados métodos de otimização global baseados em heurísticas, como algoritmos genéticos em vez de algoritmos de programação matemática [28].

Quando se formula um problema de otimização estrutural tem de se ter em conta: a função objetivo (peso da estrutura, custos), as variáveis de projeto (dimensões, forma, topologia), constrangimentos (deslocamentos, tensões, frequências) e os limites laterais das variáveis de projeto.

A maioria dos métodos desenvolvidos para resolver problemas de otimização procuram iterativamente no espaço das variáveis de projeto o ponto que minimiza a função objetivo verificando simultaneamente os constrangimentos. Essa pesquisa é feita com base no valor da função objetivo e dos constrangimentos e também dos seus gradientes em relação às variáveis de projeto.

A necessidade de calcular gradientes exige a continuidade das funções utilizadas, o que é uma limitação em alguns problemas. Por outro lado, os métodos baseados em gradientes têm muita dificuldade em lidar com funções que apresentem mínimos locais.

Um dos problemas em que não existe continuidade das funções corresponde ao problema de otimização com variáveis discretas.

Existem três classes de otimização, e na conceção do projeto deve-se escolher qual delas utilizar: otimização dimensional, otimização geométrica e otimização topológica [32].

Otimização dimensional

A otimização dimensional utiliza como variável de projeto um parâmetro dimensional de um elemento estrutural. São típicos desta classe de problemas, as variáveis associadas às dimensões transversais da secção de uma viga ou à espessura de uma placa.

Otimização de forma

A otimização de forma tem como objetivo definir a posição no espaço da fronteira de um domínio, de forma a melhorar uma função custo e as restrições mecânicas de projeto. Variáveis associadas à posição no espaço dos nós da malha de elementos finitos são típicos desta classe de problemas.

Otimização topológica

A otimização topológica tem o objetivo de definir da melhor forma possível a distribuição de material num domínio pré-determinado.

2.5. Algoritmos Genéticos

Os algoritmos genéticos são uma heurística de otimização de espectro global, integrada num cada vez mais vasto grupo de técnicas reconhecidas como Sistemas Evolucionários [29].

Os algoritmos genéticos procuram reproduzir computacionalmente o processo de seleção natural das espécies e utilizam terminologia da genética. Este método, como se pode ver na figura 2.14, utiliza representações binárias das possíveis soluções de um problema e transformações destinadas a aperfeiçoar essas soluções de forma a atingir a solução ótima [29].

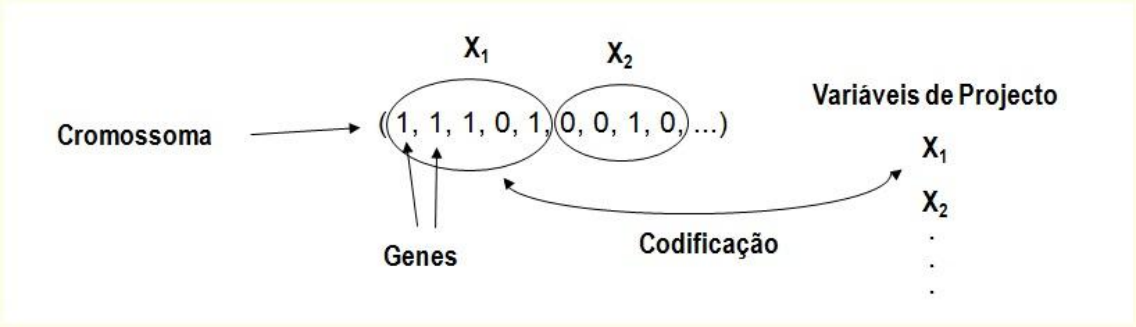


Figura 2.14 – Codificação de variáveis num algoritmo genético

Neste algoritmo as variáveis do problema são representadas como genes num cromossoma, também denominado indivíduo. O conjunto de cromossomas constituindo a população de uma determinada geração é combinada através de operadores para dar origem à população da geração seguinte, que contém indivíduos melhor adaptados, de acordo com uma função de mérito [28].

A aplicação de um algoritmo genético envolve: a codificação das variáveis de projeto, a definição da função de mérito e a definição de operadores que alterem o conteúdo dos cromossomas.

Os operadores que alteram o conteúdo dos cromossomas são:

Seleção – escolhe os indivíduos de uma geração que deverão fazer parte da geração seguinte.

Cruzamento – combina os genes de dois cromossomas pais para dar origem a dois cromossomas filhos distintos dos progenitores.

Mutação – altera de forma aleatória os genes de um cromossoma.

A figura 2.15 representa a implementação de um algoritmo genético.

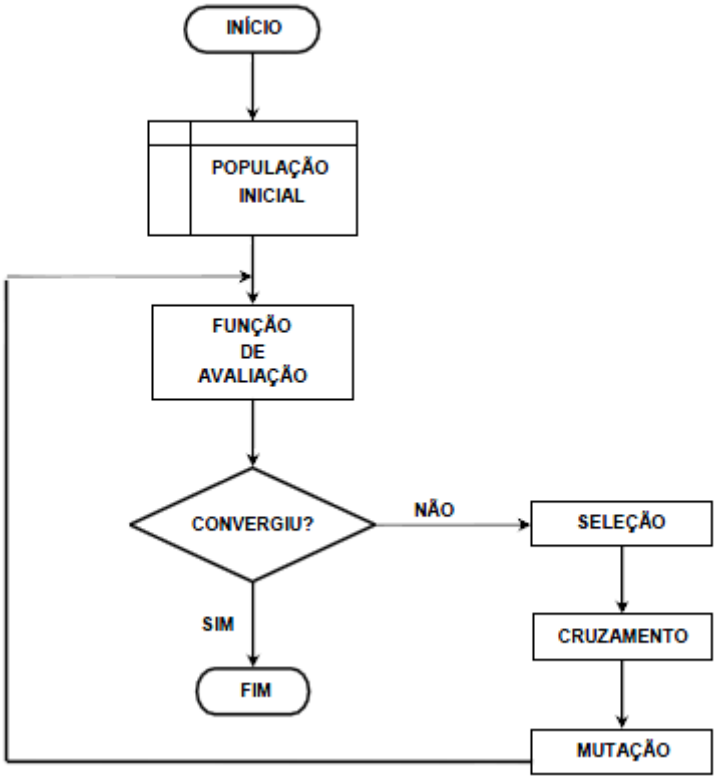


Figura 2.15 – Implementação de um algoritmo genético [28]

A implementação de um algoritmo genético é feita da seguinte forma:

1) Gerar aleatoriamente uma população de cromossomas (possíveis soluções para o problema);
2) Calcular a aptidão de cada indivíduo na população, utilizando a função de avaliação;
3) Selecionar os cromossomas da população atual que formarão descendentes da próxima geração. Aplicar operadores de cruzamento e mutação sobre cromossomas selecionados pra criar a próxima geração de cromossomas;
4) Substituir a população atual pela nova população gerada;
5) Se o método convergir, o processo termina e devolve o melhor indivíduo gerado. Caso não convirja, volta ao passo 2).

Os algoritmos genéticos neste trabalho serão utilizados no exemplo da Estrutura de 6 barras, por forma a otimizar a estrutura de seis barras. Para tal recorre-se à *toolbox* de algoritmos genéticos do MATLAB.

Capítulo3

Utilização de Redes Neuronais Artificiais no MATLAB

A componente prática da presente dissertação recorre ao programa MATLAB. Pretende-se por isso, explicar neste capítulo, o funcionamento das interfaces gráficas do MATLAB que foram utilizadas.

3.1. MATLAB

O MATLAB é um ambiente de computação e desenvolvimento de aplicações, totalmente integrado e orientado para levar a cabo projetos que impliquem cálculos matemáticos complexos e a sua visualização gráfica. O MATLAB integra análise numérica, cálculo matricial, processamento de sinais e visualização gráfica, num meio onde os problemas e as suas soluções são expressas de modo semelhante àquele em que se escreviam tradicionalmente, sem necessidade de fazer uso de técnicas de programação tradicionais [33].

O nome de MATLAB resultou da contração das palavras "MATrix LABoratory" e foi inicialmente concebido para proporcionar fácil acesso às bibliotecas LINPACK e EISPACK, que representavam duas das mais importantes em computação e cálculo matricial [33].

O MATLAB é um sistema de trabalho interativo baseado na representação matricial. O programa permite desenvolver algoritmos de um modo muito mais rápido do que com linguagens de programação tradicionais como Fortran, Basic ou C e por esse facto é eficientemente usado na resolução numérica de problemas, num tempo muito menor [33].

As utilizações mais frequentes desta ferramenta encontram-se nas áreas de computação e cálculo numérico tradicional, prototipagem, algoritmia, teoria do controle automático, estatística, análise de séries temporais para o processamento digital de sinais [33].

O MATLAB dispõe também de um amplo conjunto de programas de apoio especializados, denominados *Toolboxes* e de interfaces gráficas, que estendem significativamente o número de funções incorporadas no programa principal. Estas interfaces gráficas cobrem praticamente todas as áreas principais no mundo da engenharia, destacando entre elas a interface gráfica de processamento de imagens, sinais, controle robusto, estatística, análise financeira, cálculo matemático simbólico, redes neuronais, lógica difusa, identificação de sistemas, simulação de sistemas dinâmicos, algoritmos genéticos, etc. [33].

No presente trabalho são utilizadas duas interfaces gráficas de redes neuronais artificiais, *nftool* e *nntool*, as quais são explicadas nas secções 3.1.1 e 3.1.2 respetivamente.

Por forma a avaliar a capacidade das redes neuronais artificiais do MATLAB em aproximar uma função não linear, e demonstrar como funcionam as interfaces gráficas, utiliza-se uma função de teste apresentada em [7] expressa por:

$$F(x,y) = 0.3 + (2\sin(x)\cos(y) + \sin(xy))/6 \tag{3.1}$$

Com $x \in [3.5,5.5]$ e $y \in [2.0,4.0]$. A figura 3.1 representa graficamente a função.

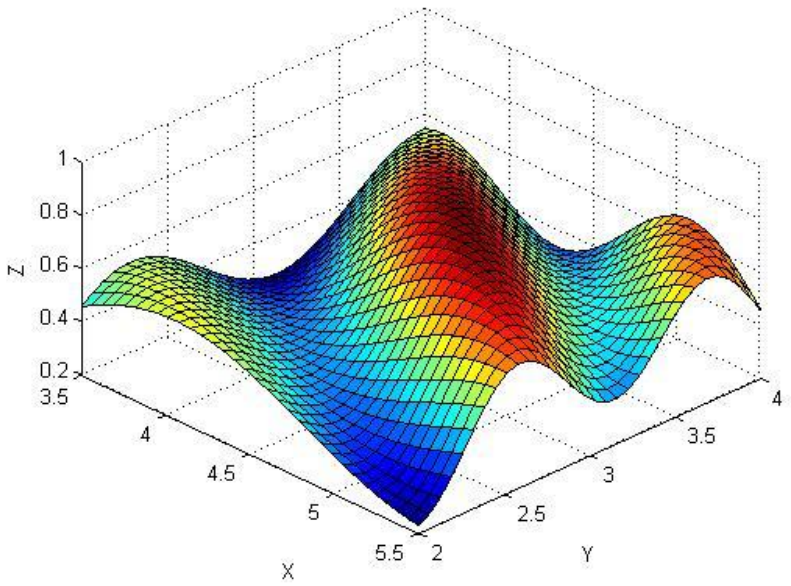


Figura 3. 1 - Representação gráfica da função de teste

Para treinar a rede são necessários valores de entrada e de saída da rede, utilizando-se um programa criado em MATLAB para gerar estes valores. Foi gerado um conjunto de 196 valores diferentes, considerando 14 pontos igualmente espaçados de cada variável de entrada, utilizando este conjunto para treino, e um conjunto de 1024 valores diferentes, considerando 32 pontos igualmente espaçados de cada variável de entrada, utilizando este conjunto para testar a rede. Neste caso não é necessário usar o hipercubo latino porque são só duas variáveis.

De salientar que de acordo com o manual de utilizador do MATLAB, é aconselhável que os valores de entrada e de saída sejam normalizados para que a rede dê resultados aceitáveis [34]. Desta forma este mesmo programa, normaliza tanto os parâmetros de entrada como os parâmetros de saída.

Consideraram-se várias redes neuronais artificiais, mantendo constante o número de neurónios na camada de entrada e de saída, s^0 e s^2, respetivamente iguais a 2 e 1, variando apenas o número de neurónios na camada intermédia, s^1, utilizando-se redes com 1, 6, 12, 18, 24, 30, 36 neurónios na camada intermédia.

Por forma a treinar a rede, recorre-se às interfaces gráficas de redes neuronais artificiais do MATLAB, explicando-se de seguida como se utiliza cada uma delas.

3.1.1. nftool

Uma das interfaces gráficas utilizadas para a criação de redes neuronais artificiais em ambiente MATLAB é a *neural network fitting tool (nftool)* [34].

Esta interface gráfica permite treinar uma rede, tendo já definida a utilização de uma rede progressiva multicamada com três camadas (*multilayer feedforward network*), como a representada na figura 2.11, mas com menos uma camada, e treino supervisionado através do algoritmo de Levenberg-Maquardt [34].

Explica-se de seguida o como utilizar esta interface gráfica:

1) Na janela de comados do MATLAB, inserir ***nftool***, por forma a abrir a interface gráfica. Surge a janela (Figura 3.2) que introduz a interface gráfica, onde é possível observar a representação da rede neuronal que se vai criar, bem como uma descrição de como é treinada a rede;

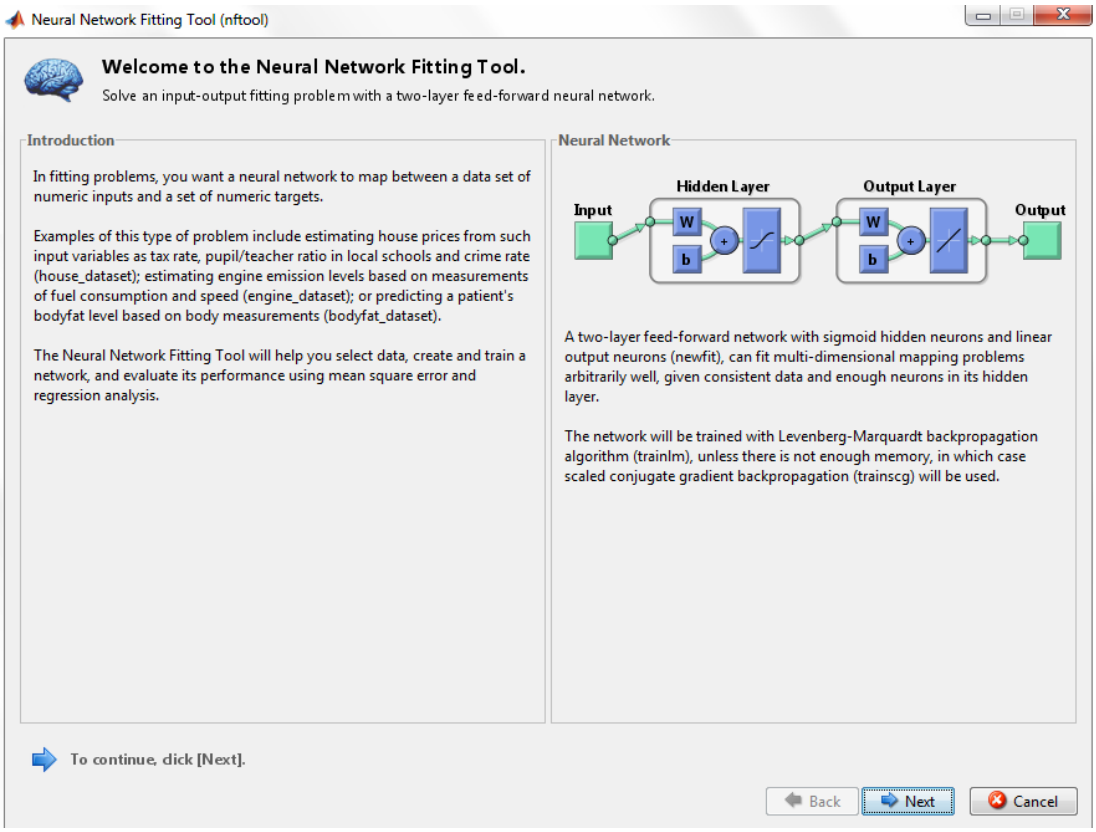


Figura 3.2 – Janela inicial da nftool

2) Clicando em ***Next***, abre-se uma janela (Figura 3.3) onde é possível selecionar os valores de entrada da rede (*inputs*) e os valores de saída da rede (*outputs*), destinados ao treino da rede a partir de matrizes carregadas anteriormente no MATLAB. Nesta janela, existe ainda a opção ***Load Example Data Set***, a partir da qual é possível carregar exemplos de demonstração disponíveis para treinar uma rede, como se mostra na figura 3.4;

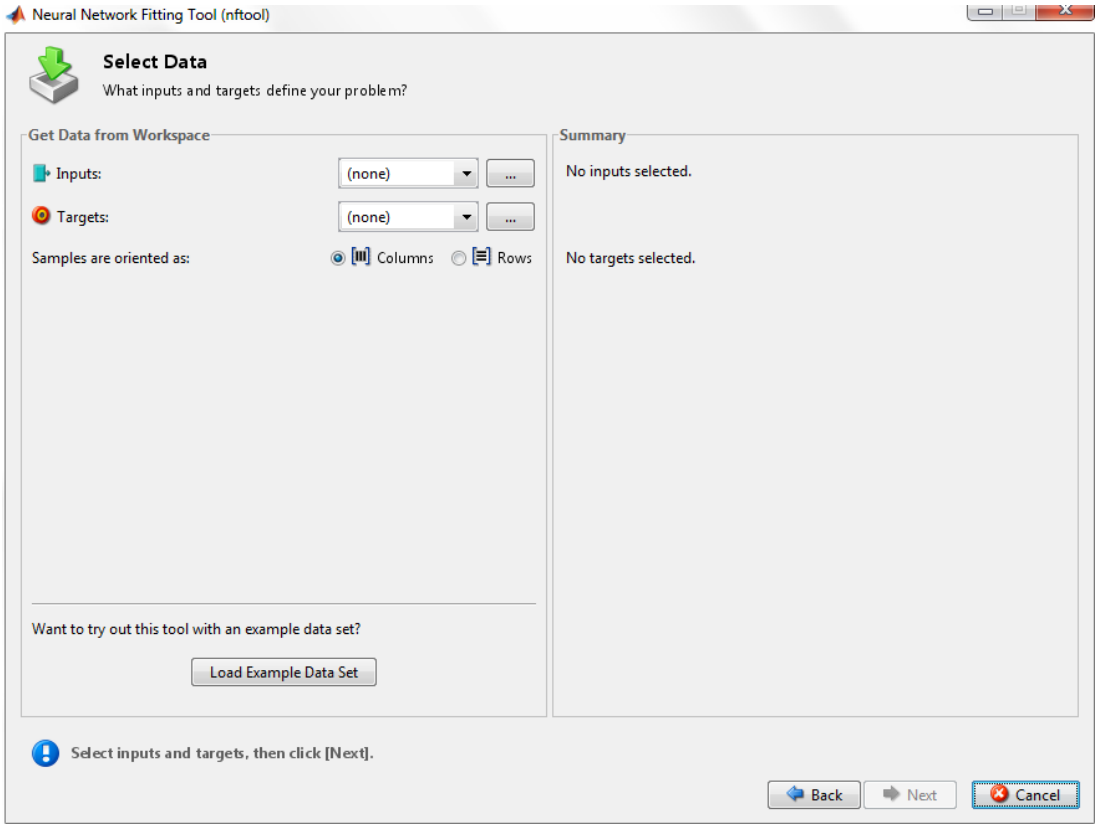


Figura 3.3 – Seleção do conjunto de treino da rede

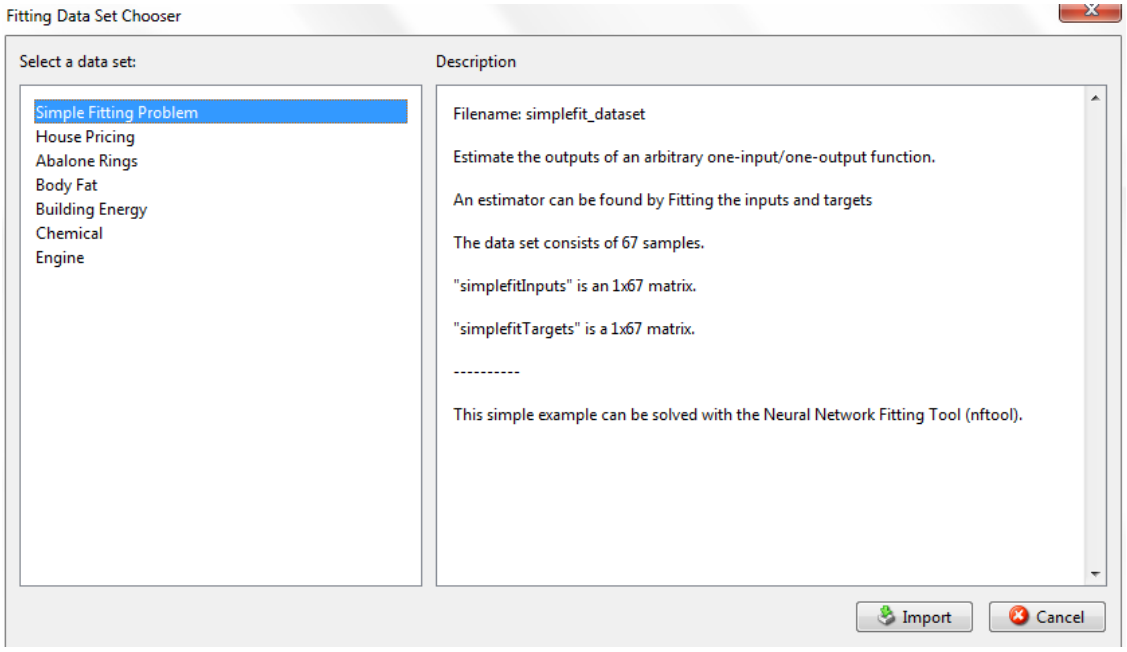


Figura 3.4 – Exemplos de conjuntos de treino

3) Após selecionar os valores de entrada e de saída da rede, carrega-se em ***Next***, e abre-se uma janela (Figura 3.5) onde são apresentadas as percentagens de valores do conjunto de treino que serão utilizadas para treino, validação e teste, sendo possível editar as

percentagens de validação e de teste. Por defeito a interface gráfica atribui 70% para treino, 15% para validação e 15% para testar, e são estes os valores utilizados neste trabalho. A validação consiste em medir o erro que a rede comete quando generaliza e permite controlar o processo do treino que termina quando este erro é suficientemente reduzido. O erro no subconjunto de teste não tem influência sobre o processo do treino e permite uma medida independente do desempenho da rede treinada;

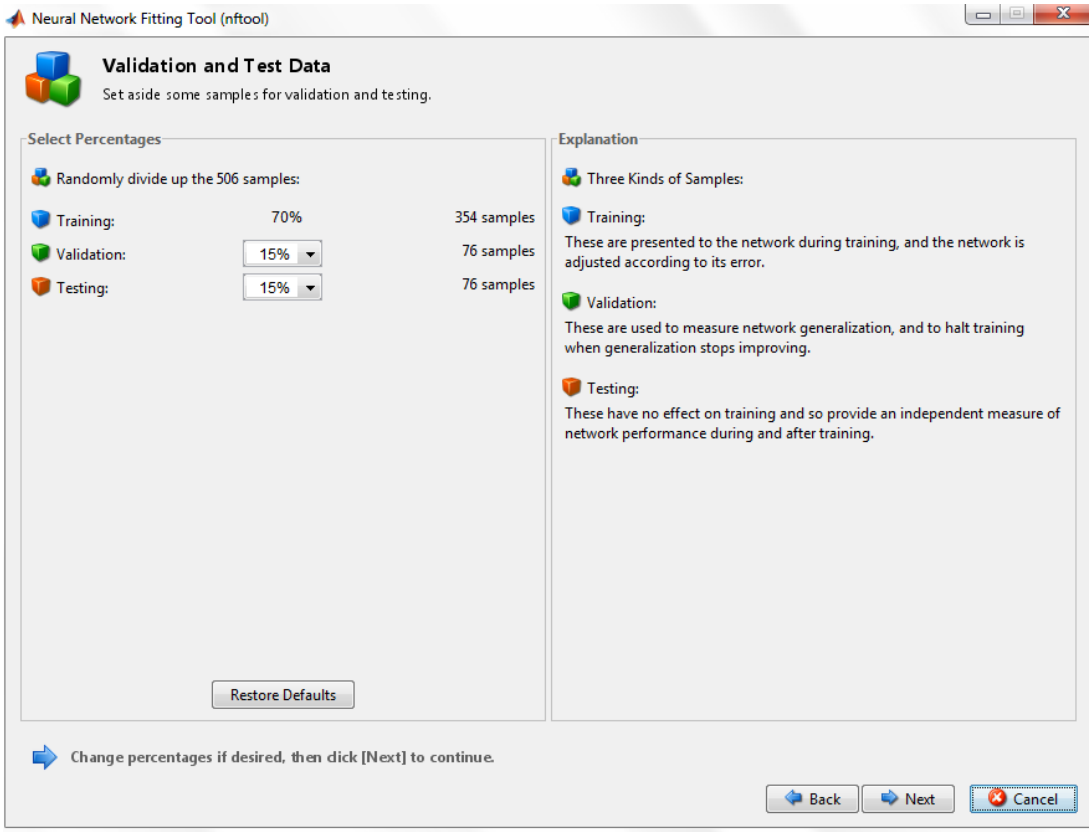


Figura 3.5 – Seleção das percentagens de valores do conjunto de treino utilizados em cada etapa

4) Clicando ***Next***, passa-se a uma janela (Figura 3.6) onde é possível editar o número de neurónios da camada intermédia;

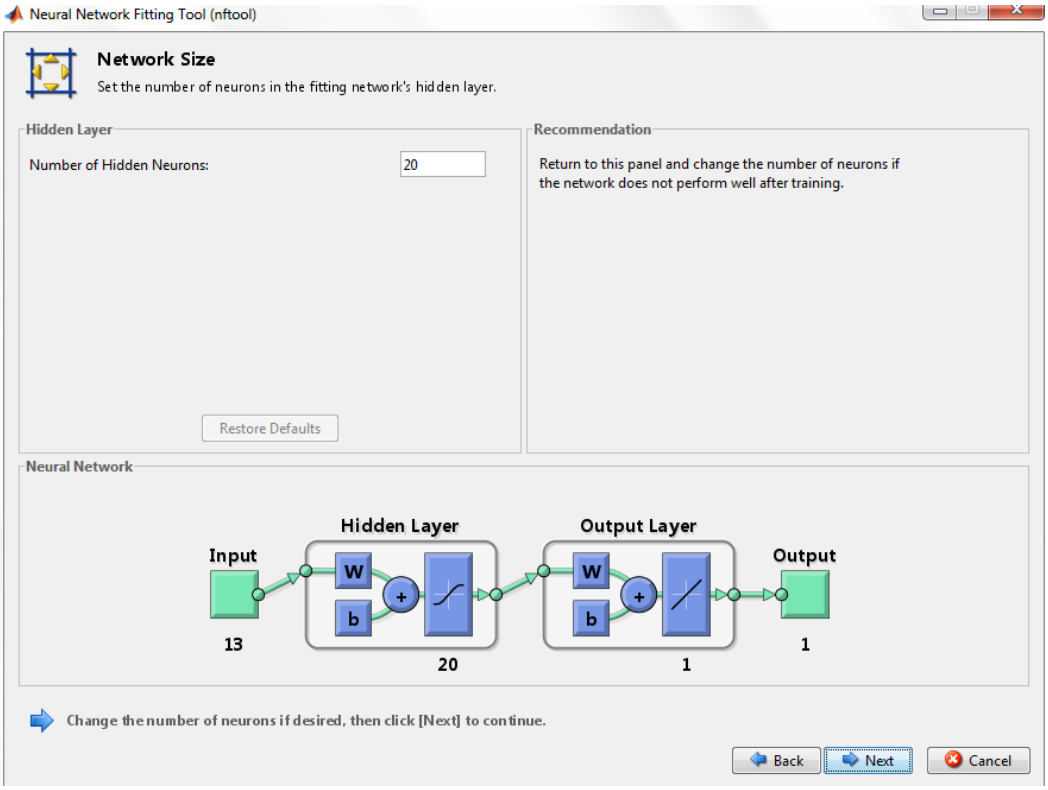


Figura 3.6 – Escolha do número de neurónios da camada intermédia

5) Clicando ***Next***, passa-se a uma janela (Figura 3.7) onde se pode carregar em ***Train*** por forma a iniciar o treino da rede;

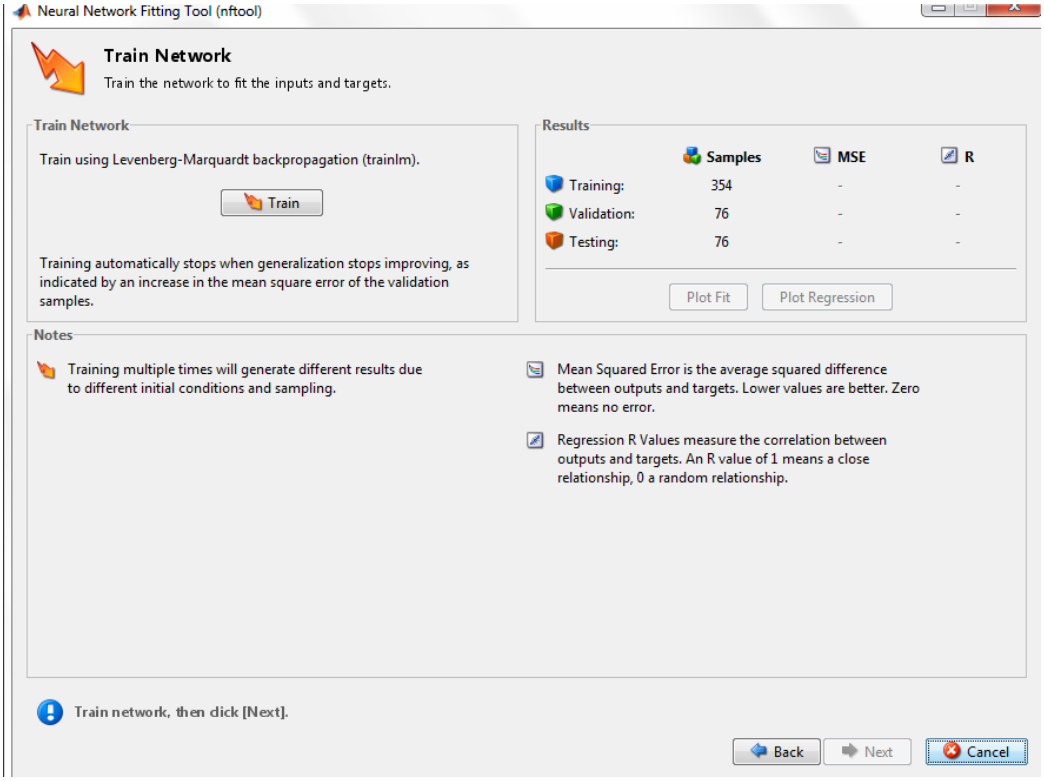


Figura 3.7 – Iniciar o treino da rede

6) Na figura 3.8 é possível observar a janela onde se vê o progresso do treino e onde é possível verificar a alteração de alguns parâmetros ao longo do tempo. Estes parâmetros têm valores limite, como se pode ver na figura, terminando o treino quando é atingido um destes valores limite. No caso da figura 3.8, foi atingido o valor máximo de *Validation Checks*. Nesta janela aparecem ainda as opções *Performance* (Figura 3.9), onde se pode ver a evolução do erro quadrático médio em cada uma das fases, calculado através das equações 2.32, 2.33 e 2.34, *Regression* (Figura 3.10) onde se mostra uma reta de regressão linear da aproximação dos pontos durante cada uma das fases;

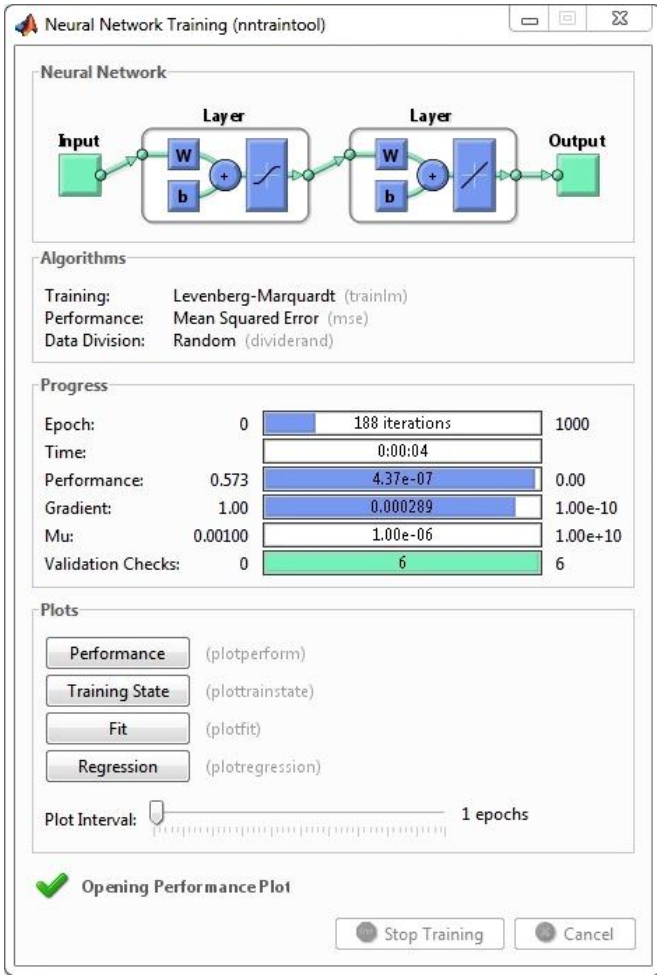


Figura 3.8 – Progresso do treino

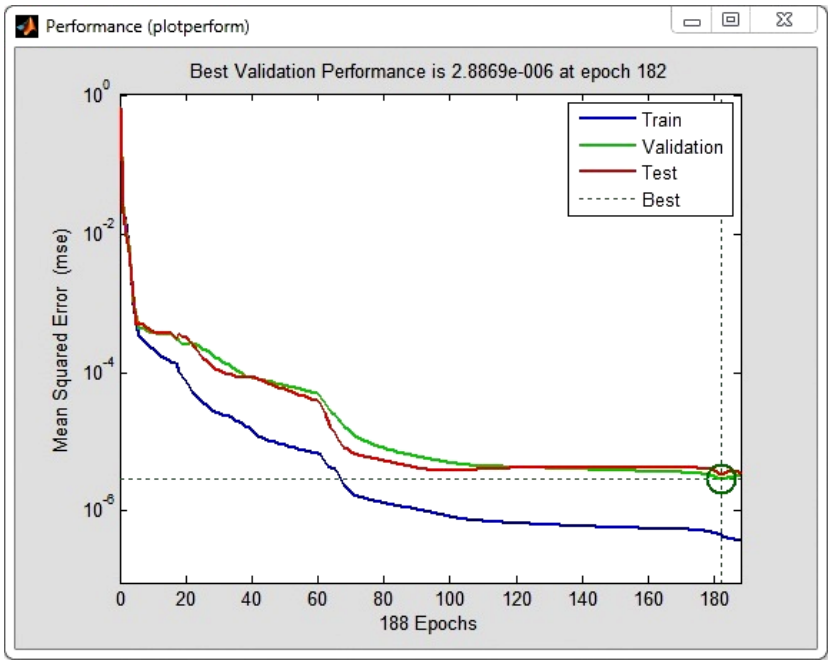


Figura 3.9 – Gráfico de *Performance*

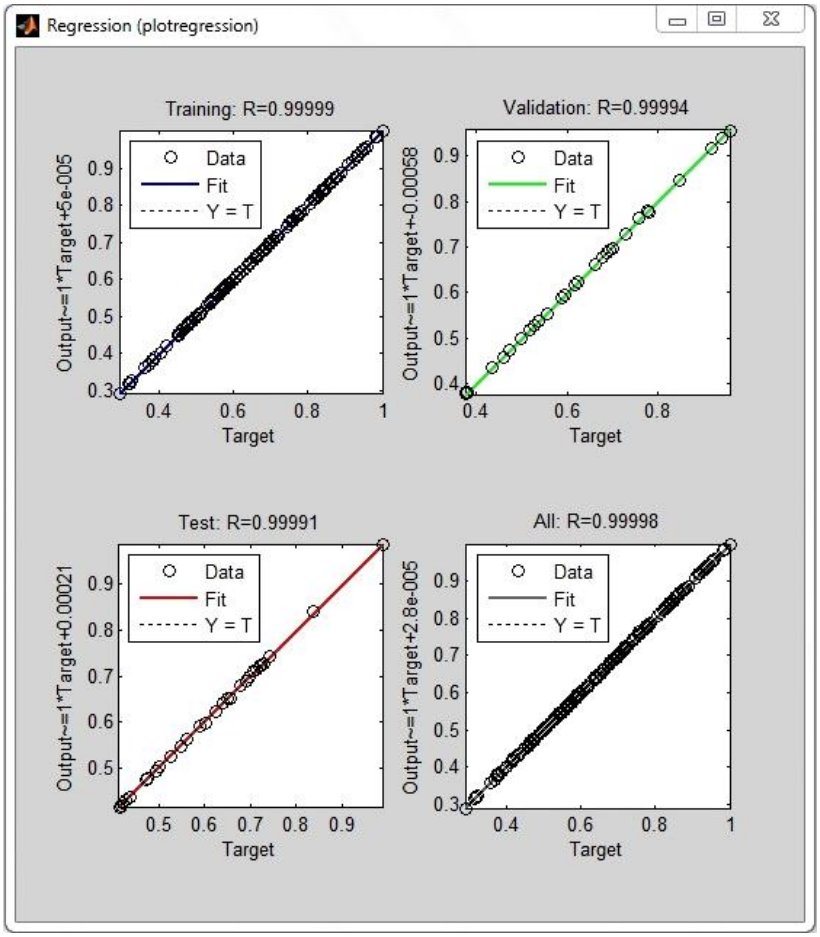


Figura 3. 10 – Gráficos de *Regression*

7) Finalizado o treino, aparece uma janela semelhante à da Figura 3.11, onde é possível voltar a treinar a rede, carregando na opção ***Retrain***. Além desta opção aparecem ainda os erros quadráticos médios relativos aos subconjuntos de treino, validação e teste;

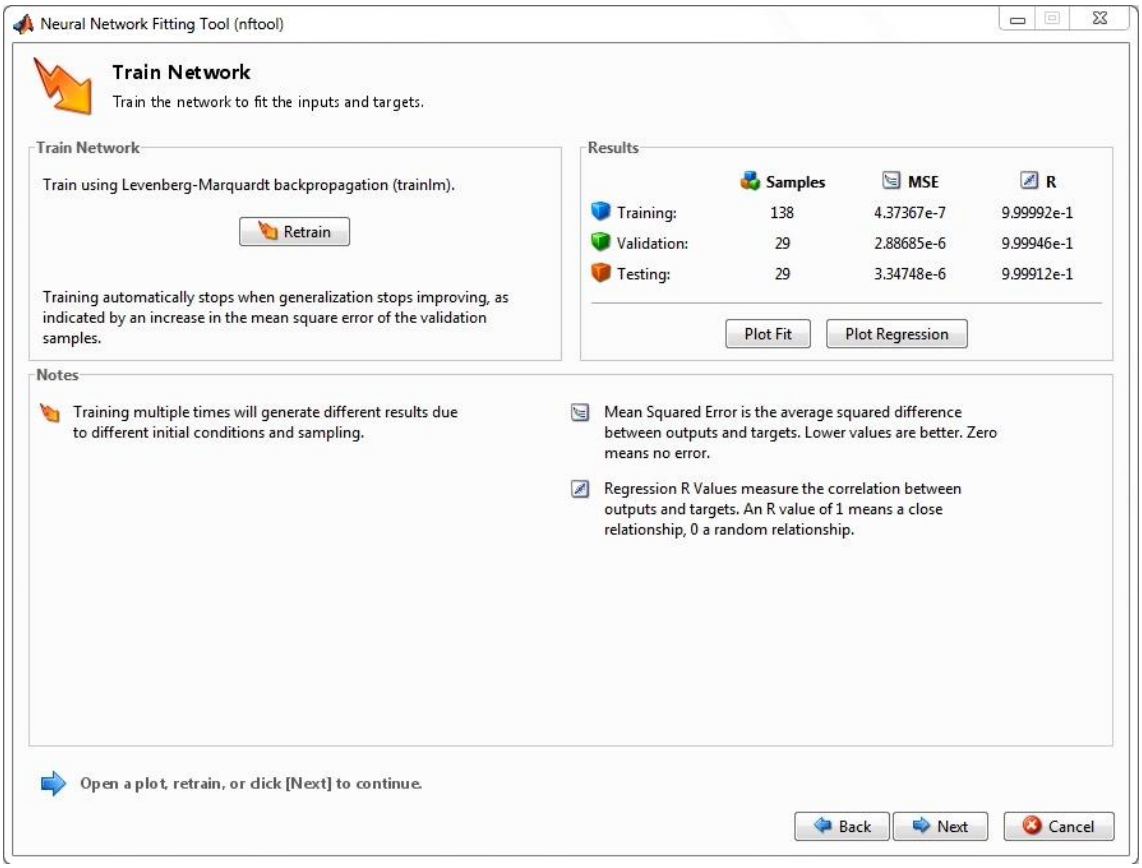


Figura 3.11 – Voltar a treinar a rede

8) Clicando ***Next***, a janela (Figura 3.12) passa a apresentar as opções *Train Again*, *Adjust Network Size*, *Import Larger Data Set*, que permitem voltar a treinar a rede (Figura 3.7), ajustar o número de neurónios da camada intermédia (Figura 3.6) e modificar o conjunto de treino (Figura 3.3), por forma a melhorar o treino da rede;

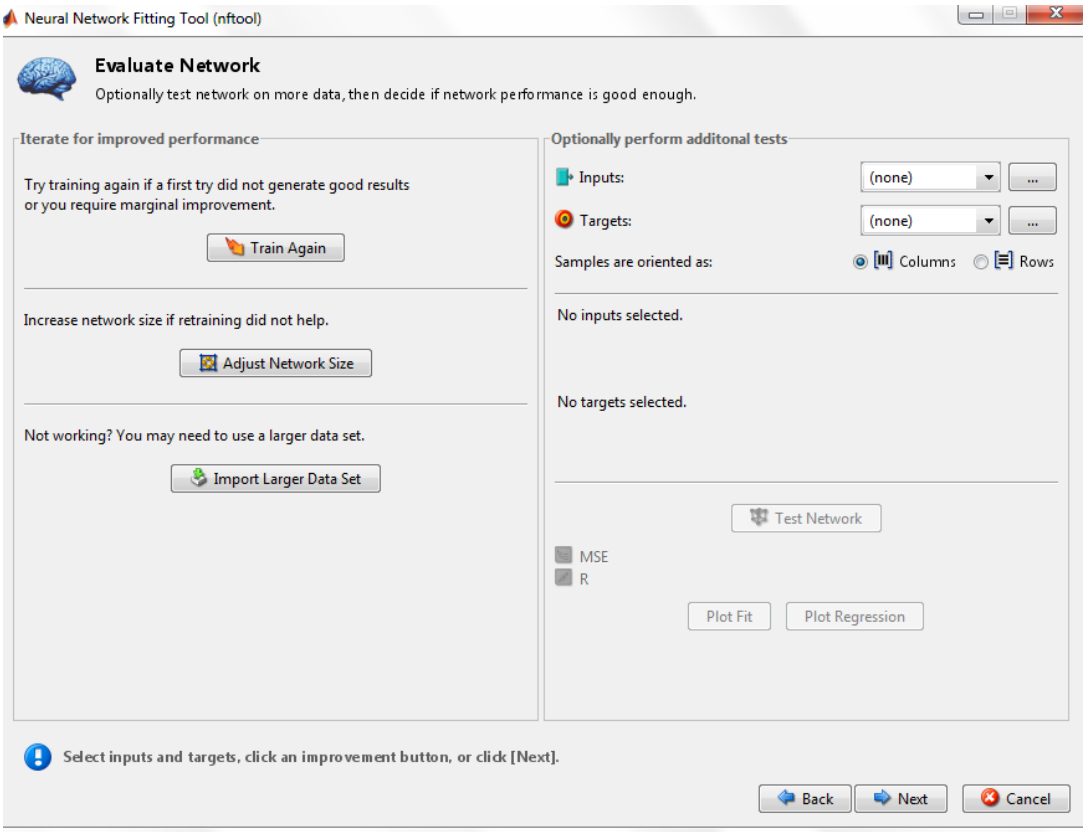


Figura 3.12 – Melhorar o treino

9) Clicando ***Next***, passa-se a uma janela (Figura 3.13) onde é possível gravar no *Workspace* do MATLAB, todos os parâmetros relativos ao treino, terminando desta forma o processo de treino da rede, permitindo posteriormente utilizar estes resultados para simular a rede. É ainda possível criar o ficheiro de MATLAB (***Generate M-File***) que permite reproduzir os resultados e resolver outros problemas e/ou um ficheiro de *Simulink* (Conjunto de diagramas que simulam o funcionamento da rede), através de ***Generate Simulink Diagram***.

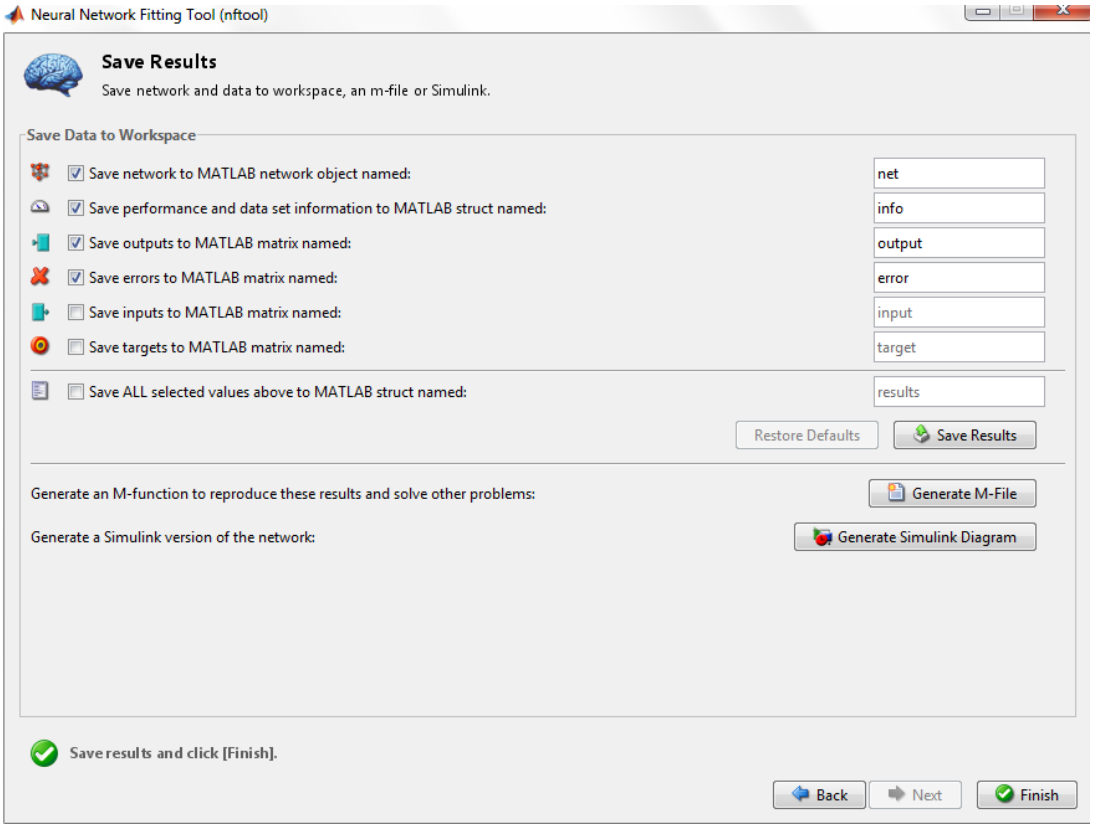


Figura 3.13 – Guardar os resultados

Repete-se este procedimento para treinar cada uma das redes com diferente número de neurónios na camada intermédia. .

Dado que os pesos iniciais das redes são valores aleatórios é possível obter resultados diferentes para os mesmos parâmetros de treino da rede, podendo ter de se treinar a rede várias vezes até se obter resultados aceitáveis. Finalizado o treino da rede, esta fica pronta para ser utilizada.

3.1.2. nntool

Outra das interfaces gráficas utilizadas para a criação de redes neuronais artificiais em ambiente MATLAB é a *nntool*, que pode ser usada em alternativa à *nftool*.

Esta interface gráfica permite treinar uma rede, permitindo a edição de vários campos, como se mostra de seguida. Para efetuar o treino procede-se da seguinte forma:

1) Na janela de comados do MATLAB, inserir ***nntool***, por forma a abrir a interface gráfica. Abre-se uma janela, como a apresentada na figura 3.14;

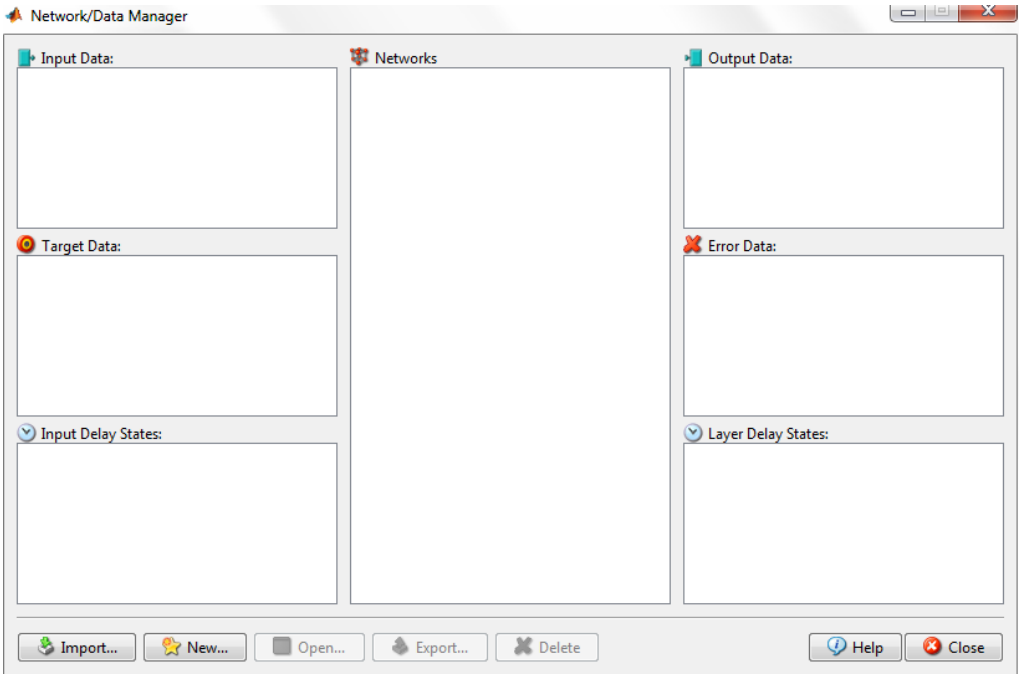


Figura 3.14 – Janela inicial da nntool

2) Carregando em ***Import***, é possível definir alguns parâmetros de rede tais como o conjunto de valores de treino e valores iniciais dos pesos da rede, ou até uma rede previamente treinada, como se pode ver na figura 3.15, podendo-se escolher onde ir buscar os valores, quais os valores e qual a sua função na rede. Neste ponto definem-se como *Input Data* os valores de entrada da rede, quer para treino quer para teste, e como *Target Data* os valores de saída da rede, para treino;

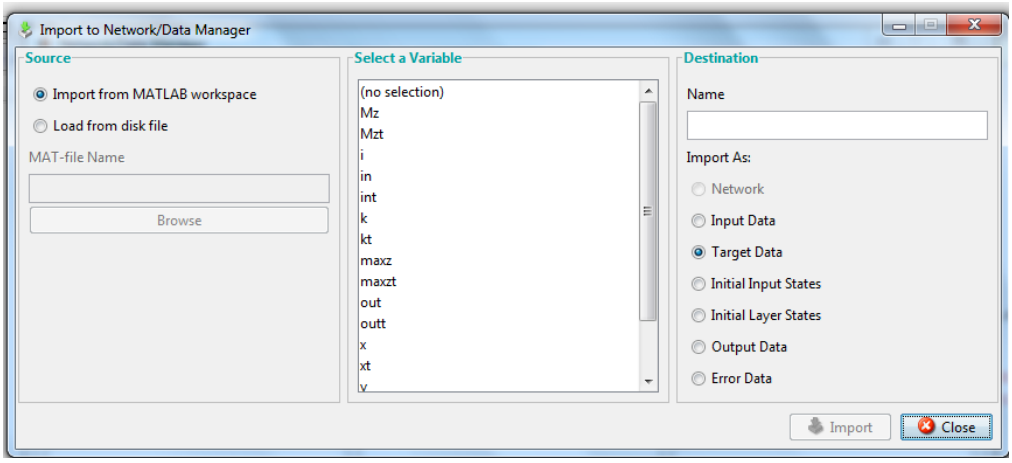


Figura 3.15 – Parâmetros da rede

3) Na janela representada na figura 3.14, existe a ainda a opção ***New…***, na qual é possível carregar, abrindo-se uma janela como a representada na figura 3.16 e na qual é possível criar a rede, definindo-se vários parâmetros como o tipo de rede, *inputs*, *outputs*, função de ativação, algoritmo de aprendizagem, número de camadas, número de neurónios, havendo várias opções possíveis para cada um dos parâmetros;

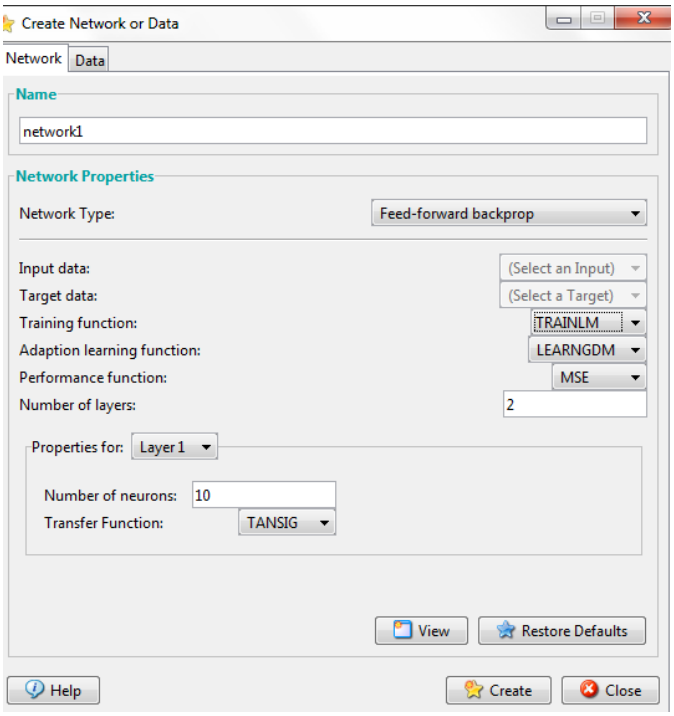


Figura 3.16 – Criar a rede

4) Após a definição dos parâmetros da rede e da criação da rede, é possível clicar na opção ***Open***, apresentada na figura 3.14, abrindo-se uma janela semelhante à apresentada na figura 3.17. Nesta janela existem vários separadores permitem treinar a rede carregando na opção ***Train*** e editar os parâmetros de treino (Figuras 3.18 e 3.19), simular a rede (Figura 3.20), adaptar melhor a rede e editar os parâmetros de adaptação (Figuras 3.21 e 3.21), reinicializar os pesos da rede (Figura 3.23) e ver/editar os pesos da rede (Figura 3.24);

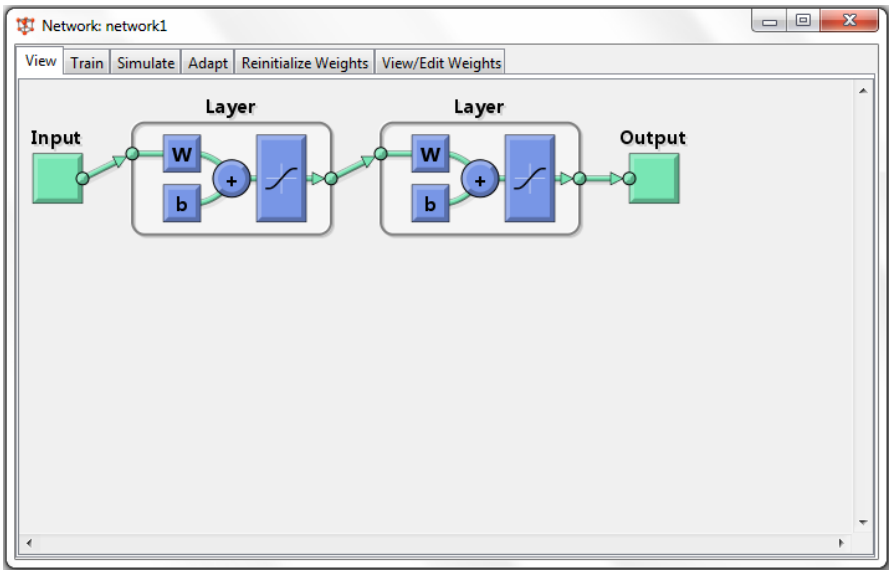


Figura 3.17 – Esquema da rede

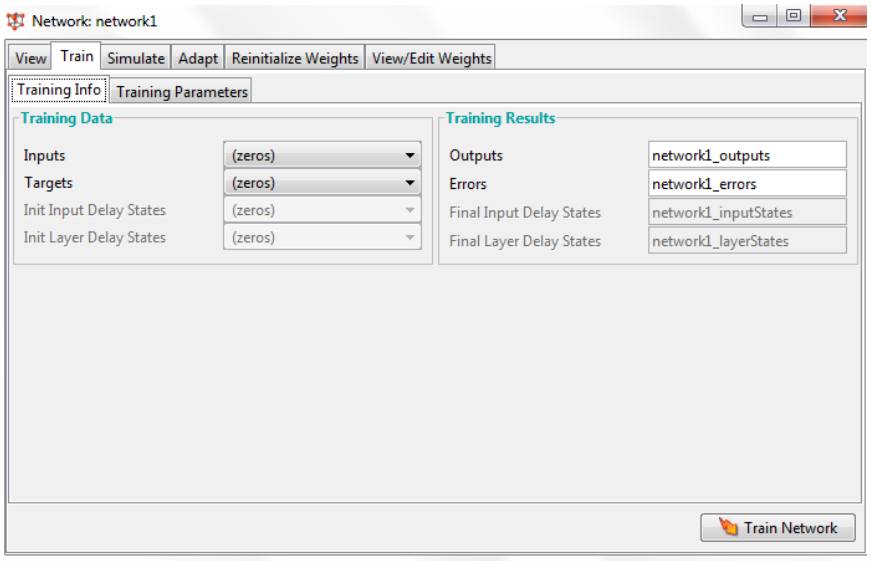


Figura 3.18 – Treino da rede

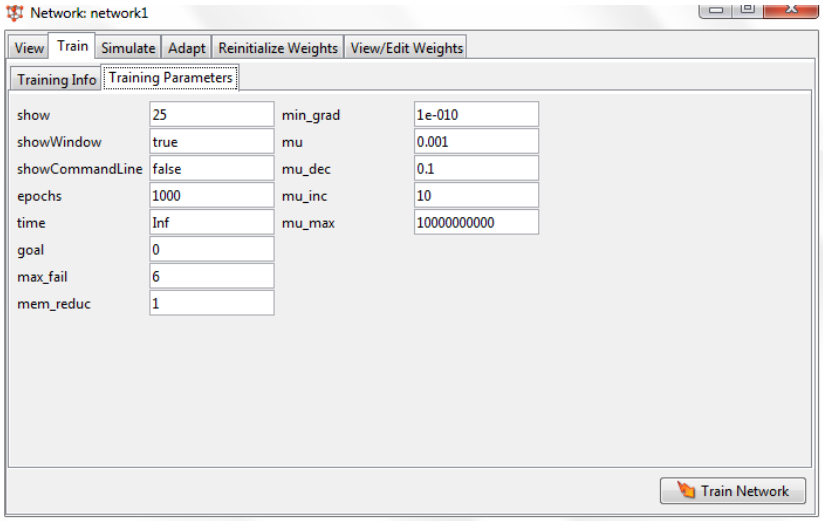


Figura 3.19 – Editar parâmetros de treino

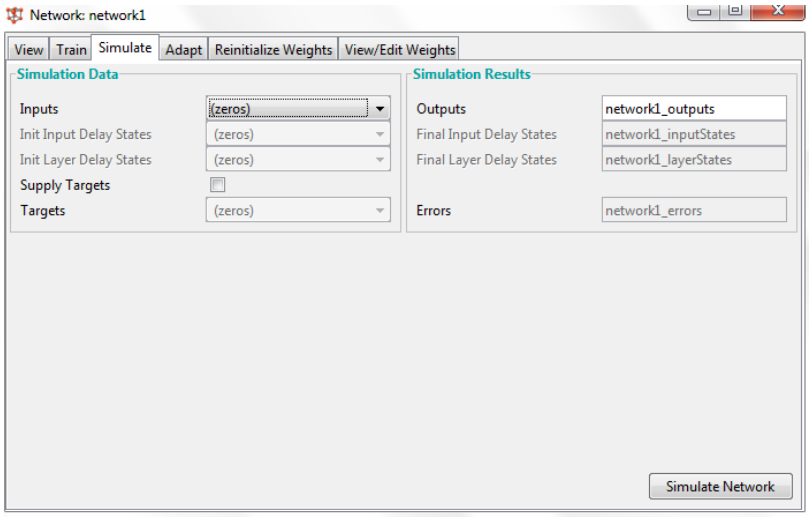


Figura 3.20 – Simular a rede

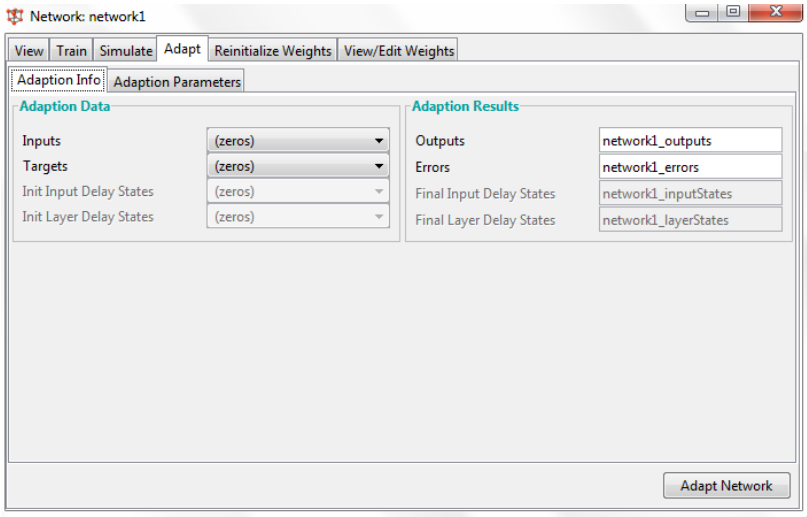


Figura 3.21 – Adaptar a rede

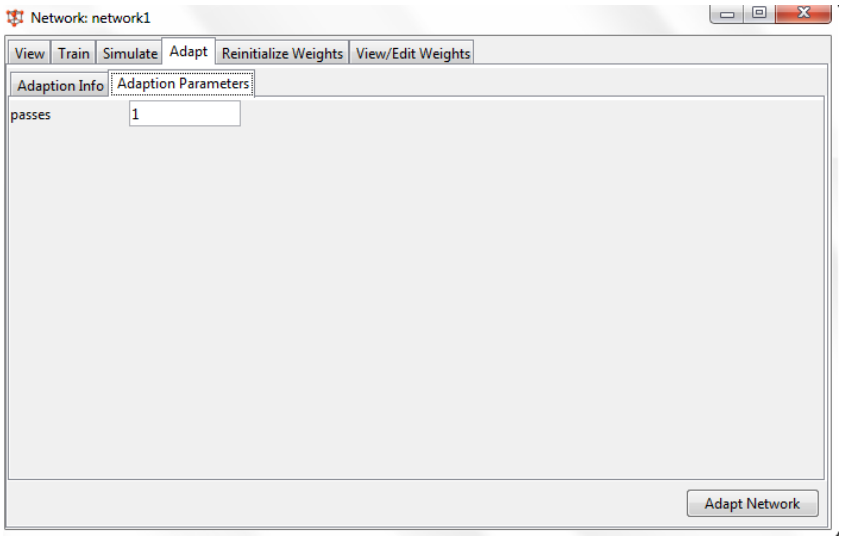


Figura 3.22 – Parâmetros de adaptação da rede

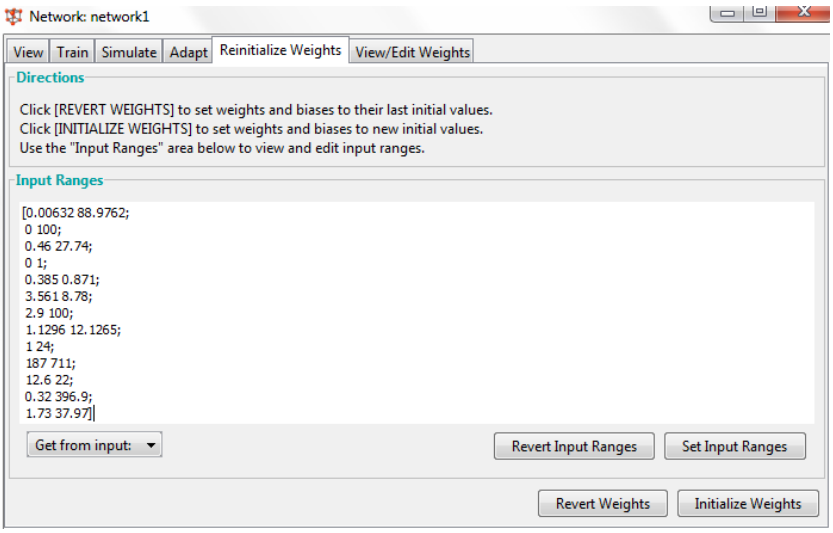


Figura 3.23 – Reinicializar os pesos da rede

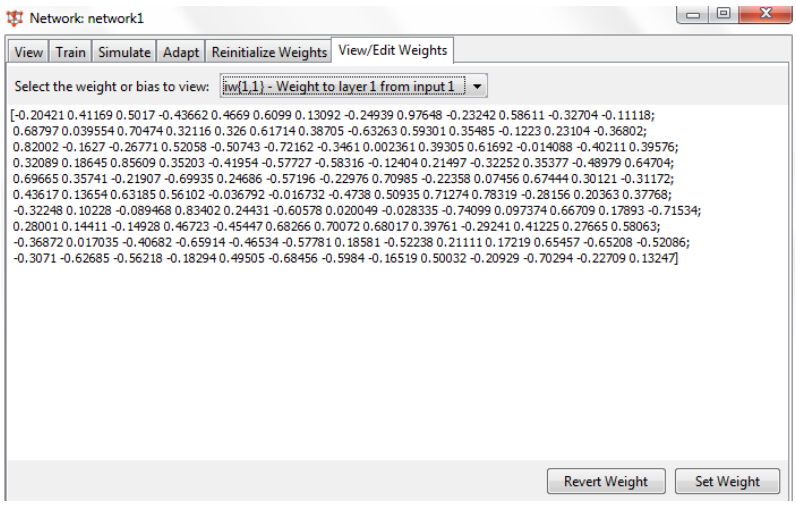


Figura 3.24 – Ver/Editar os pesos da rede

5) Treinando a rede, carregando na opção ***Train*** (Figuras 3.18 e 3.19), aparece uma janela, como a apresentada na janela ilustrada na figura 3.8. Após treinar a rede é possível fazer a simulação carregando na opção ***Simulate Network*** (Figura 3.20). Após simular aparece uma janela como a da figura 3.25;

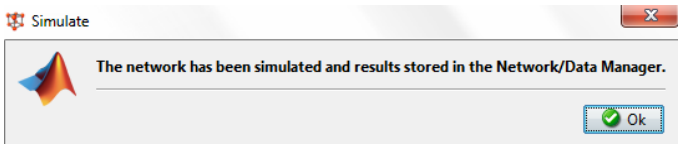


Figura 3.25 – Janela confirmando a simulação da rede

6) Tendo já todos os resultados necessários é possível gravá-los como se mostra na figura 3.26. Desta forma tem-se uma rede treinada e simulada, podendo desta forma verificar o erro da rede e utilizá-la de futuro na resolução de problemas.

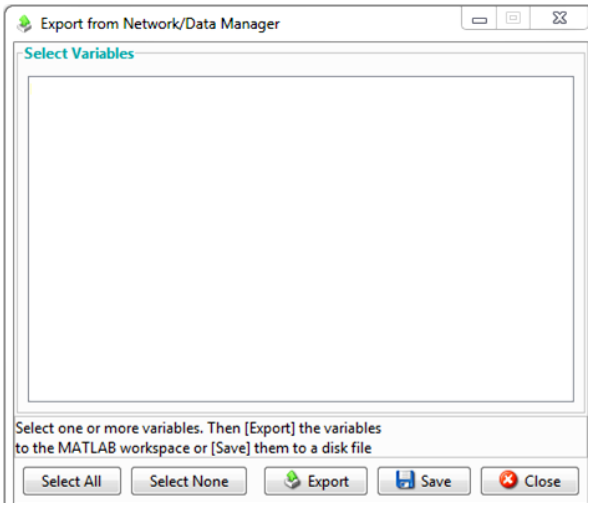


Figura 3.26 – Guardar os resultados

Repete-se este procedimento para treinar cada uma das redes com diferente número de neurónios na camada intermédia.

Dado que os pesos iniciais das redes são aleatórios é possível obter resultados diferentes para os mesmos parâmetros de rede, podendo ter de se treinar a rede várias vezes até se conseguirem resultados aceitáveis.

Para obter uma medida rigorosa da capacidade da rede treinada aproximar a função (3.1), considerou-se o erro relativo obtido pela equação (3.2):

$$\varepsilon = \left| \frac{s_{ij} - o_{ij}}{o_{ij}} \right| \tag{3.2}$$

Para $i = 1, \ldots, t$ e $j = 1, \ldots, r$, onde s_{ij} são os valores obtidos pela simulação da rede neuronal e o_{ij} são os valores reais.

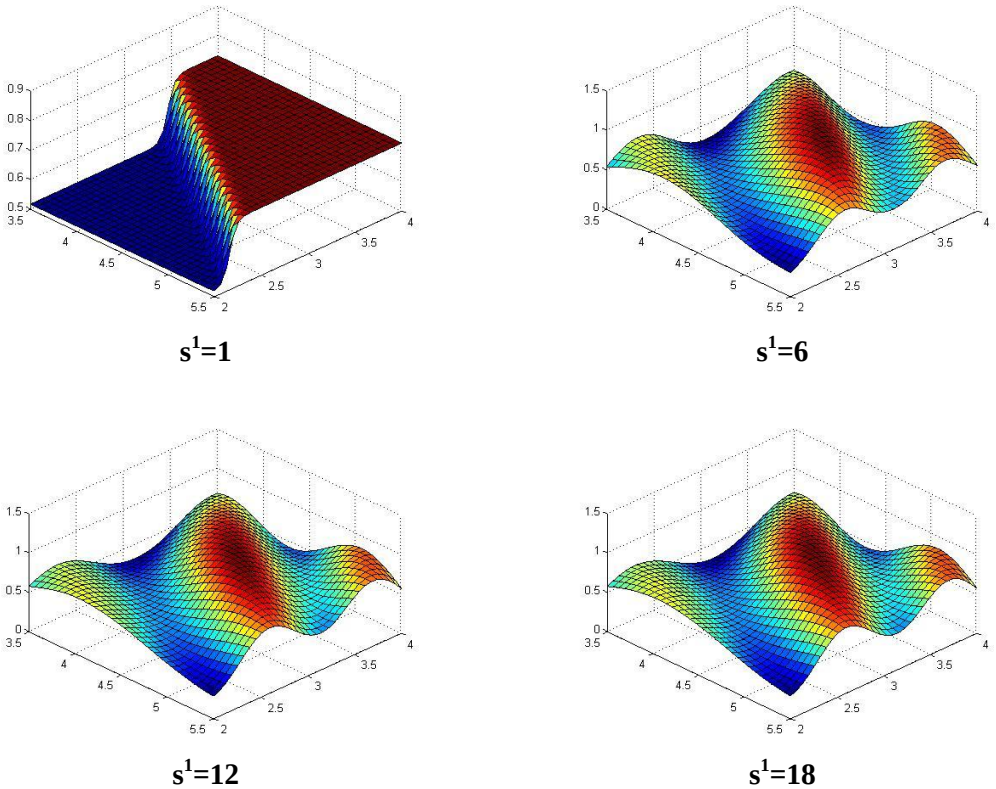
Por forma a calcular os erros relativos, máximo e médio, para todos os elementos do conjunto de teste, foi escrito um programa em linguagem MATLAB que faz este cálculo.

A interface gráfica *nntool* já permite a simulação da resposta da rede, calculando os resultados de saída da rede, para qualquer conjunto de teste, ao contrário da *nftool*. Desta forma inserem-se os valores obtidos na fase de simulação, no programa de cálculo do erro, por forma a obter os erros relativos, máximo e médio, da fase de simulação.

Após treinar várias redes para a função (3.1), obtiveram-se os erros relativos apresentados na tabela 3.1, bem como os gráficos ilustrados na figura 3.27.

Tabela 3. 1 – Erros relativos máximos e médios em função do número de neurónios da camada intermédia

s^1	Erro relativo máximo (%)	Erro relativo médio(%)
1	79	2
6	12	0,39
12	1,8	0,15
18	0,63	0,20
24	0,91	0,19
30	0,92	0,21
36	2	0,20



s^1=1 **s^1=6**

s^1=12 **s^1=18**

Figura 3. 27 – Aproximação da função de teste para diferentes neurónios na camada intermédia

De notar que a partir de 18 neurónios na camada intermédia, se verifica um aumento do erro relativo máximo, pelo que se conclui que a partir de um certo número de neurónios, a rede passa a estar bem aproximada no geral, dado o erro relativo médio não sofrer grandes alterações, mas em certos pontos esta aproximação já não é muito exata, pelo que se conclui que com o aumento do número de neurónios a rede se aproxima melhor dos valores do conjunto de treino, mas tem mais dificuldade em aproximar valores que não façam parte deste conjunto.

Capítulo4

Estrutura de 6 Barras

Após se terem obtido bons resultados ao utilizar o MATLAB para gerar redes neuronais artificiais, considera-se agora o exemplo de uma estrutura analisada em [7] e em [35].

A estrutura considerada é uma treliça de 6 barras, simplesmente apoiada, com uma carga *P* aplicada no nó 3 e tensão de cedência do material, aleatórias. A figura 4.1 ilustra a geometria inicial da estrutura em estudo.

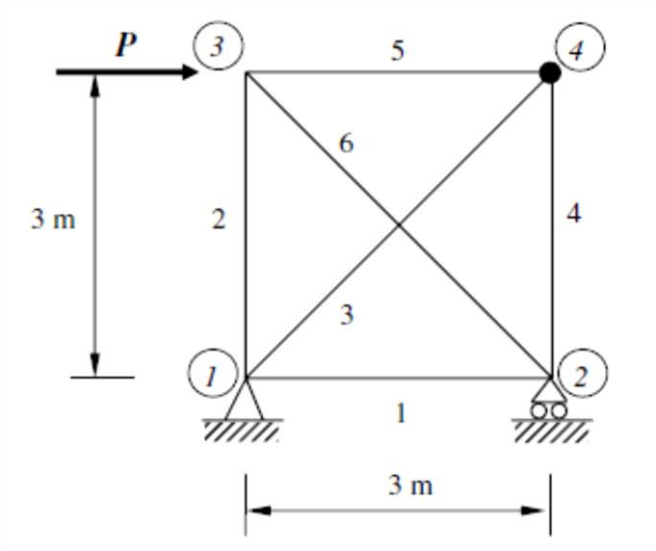


Figura 4. 1 – Treliça de 6 barras [7]

Pretende-se fazer a otimização da massa da estrutura, utilizando constrangimentos de fiabilidade.

As coordenadas *X* e *Y* do nó 4 permanecem sempre iguais, sendo a coordenada considerada como variável de projeto – *nx*. A área de secção de cada barra da treliça, *A*, é igual para todas as barras, sendo também esta uma das variáveis de projeto. Os limites das variáveis são especificados na tabela 4.1.

Tabela 4. 1 – Limites das variáveis de projeto

Variável de projeto	Limite inferior	Limite Superior
nx (m)	2.0	4.0
A (m^2)	1.0 x 10^{-4}	4.0 x 10^{-4}

A massa da treliça é otimizada considerando como variáveis aleatórias, a tensão de cedência do material, σ_c e a força aplicada no nó 3, *P*. Os parâmetros estatísticos destas variáveis encontram-se ilustrados na tabela 4.2.

Tabela 4. 2 – Parâmetros estatísticos das variáveis aleatórias

Variável	Tipo de distribuição	Média	Desvio padrão	Coeficiente de variação
P (kN)	Normal	30	3	0,10
σ_c (MPa)	Normal	172	8,6	0,05

Considera-se o módulo de Young, E = 206 GPa e a massa específica do material, ρ = 7800 kg/m^3.

A otimização é feita considerando que o colapso da estrutura ocorre quando pelo menos uma das barras falhar. Ou seja, trata-se de um sistema em série, no qual a falha de um dos elementos, implica uma falha no sistema. Desta forma são então consideradas seis funções de estado limite, uma para cada barra, sendo definidas por:

$$g_i = \sigma_c - |Ti| \tag{4.1}$$

Onde *Ti* é a tensão na barra *i*.

Assume-se que a probabilidade da barra falhar, probabilidade de colapso p_c, não deve exceder p_c = 0,001, que corresponde a um índice de fiabilidade β = 3,090, tal como em [7] e [35].

Para o cálculo da probabilidade de colapso do sistema, recorre-se ao método de Monte Carlo associado a uma rede neuronal.

Assim sendo, utiliza-se a interface gráfica do MATLAB, *nftool*, para treinar algumas redes neuronais, com diferente número de neurónios na camada intermédia, escolhendo-se depois a mais eficiente para ser utilizada para gerar valores das tensões por forma a calcular a probabilidade de colapso através do método de Monte Carlo de forma eficiente.

Redes neuronais

Várias redes neuronais artificiais foram consideradas, mantendo constante o número de neurónios na camada de entrada, s^0 igual a 3, correspondendo às variáveis de projeto, *nx* e *A* e à variável *P*. Os limites inferior e superior para a variável *P*, são respetivamente iguais à média menos três desvios padrão e a média mais três desvios padrão, tendo em conta os dados apresentados na tabela 4.1. Também o número de neurónios da camada de saída, s^2, é igual ao número de funções de estado limite mais a massa, mas, como as tensões nos elementos 1 e 2 são sempre iguais e o mesmo acontece nos elementos 4 e 5, apenas 5 neurónios são necessários para a camada de saída variando apenas o número de neurónios na camada intermédia, s^1, treinando-se redes com 15, 20, 25, 30, 35, 40 neurónios na camada intermédia.

As tensões nas barras, bem como a massa, são calculadas pelo método dos elementos finitos, utilizando o programa ANSYS.

Foi gerado um conjunto de 512 valores diferentes, considerando 8 pontos igualmente espaçados de cada variável de entrada, *nx*, *A* e *P*, utilizando este conjunto para treino, e um conjunto de 2744 valores diferentes, considerando 14 pontos igualmente espaçados de cada variável de entrada, *nx*, *A* e *P*, utilizando este conjunto para teste da rede.

Após o treino das redes obtiveram-se os erros relativos máximos definidos pela equação 3.2 e são apresentados na tabela 4.3.

Tabela 4. 3 – Erros relativos máximos

Nº neurónios na camada intermédia	Erros relativos máximos (%)				
	Massa	**Tensão1,2**	**Tensão 3**	**Tensão4,5**	**Tensão6**
15	0,11	0,91	1,09	0,97	1,22
20	0,12	0,43	0,62	0,62	0,71
25	0,09	0,55	0,48	0,58	0,50
30	0,04	0,31	0,37	0,34	0,59
35	0,07	0,28	0,33	0,31	0,41
40	0,22	0,25	0,28	0,41	0,68

A rede com 30 neurónios na camada intermédia é a que apresenta melhores resultados, sendo por isso a escolhida para a fase de otimização.

Probabilidade de colapso

Sendo a probabilidade de colapso pretendida, da ordem de grandeza de 10^3, um número de simulações de Monte Carlo igual a 10^5 é suficiente.

Foi criado um programa em linguagem MATLAB, utilizando os valores determinísticos: nx = 2,359 (m), e A = 1,557 x 10^{-4} (kN), e gerando valores das variáveis aleatórias P e σ_c, de acordo com os parâmetros estatísticos da tabela 4.1. Utilizando a função *normrnd* do MATLAB que gera números aleatórios com distribuição normal e a média e o desvio padrão especificados, foram gerados 10^5 conjuntos de valores independentes. Através da rede neuronal obtiveram-se as tensões nas barras e calculou-se a função de estado limite para cada barra. Através do método de Monte Carlo calculou-se uma probabilidade de colapso da treliça, p_c = 0,001, semelhante à obtida em [7].

Otimização

Para realizar a otimização da treliça de 6 barras recorre-se à interface gráfica de algoritmos genéticos do MATLAB, a GAtool, e cria-se uma função para calcular o mérito correspondente à massa da treliça, penalizando essa massa quando a probabilidade de colapso do sistema ultrapassa o valor admissível. A probabilidade de colapso é calculada recorrendo ao método de Monte Carlo associado às redes neuronais como explicado anteriormente. Realizou-se a otimização com constrangimento de fiabilidade, utilizando um algoritmo genético com 24 genes para cada cromossoma (12 para cada variável) e uma população de 48 cromossomas.

Tabela 4. 4 – Resultados da otimização estrutural com constrangimentos de fiabilidade

Variável	Burton e Hajela [35]	Cardoso et al. [7]	MATLAB
***nx* (m)**	2,366	2,359	2,267
***A* (m^2)**	1,557 x 10^{-4}	1,563 x 10^{-4}	1,570 x 10^{-4}
Massa (Kg)	22,450	22,512	22,512

Os resultados obtidos são apresentados na tabela 4.4, sendo comparados com os obtidos em [7] e em [35]. Dada a similaridade dos valores obtidos com os apresentados nos artigos de referência, conclui-se que os valores são satisfatórios, comprovando-se desta forma a qualidade das metodologias e das ferramentas.

Capítulo5

Placa Reforçada

Neste exemplo avalia-se a segurança de um navio da classe Aframax, projetado de acordo com as Common Structural Rules for Double Hull Oil Tankers, que foi estudado em [9] e em [36]. Um corte transversal da secção mestra do navio é ilustrado na figura 5.1.

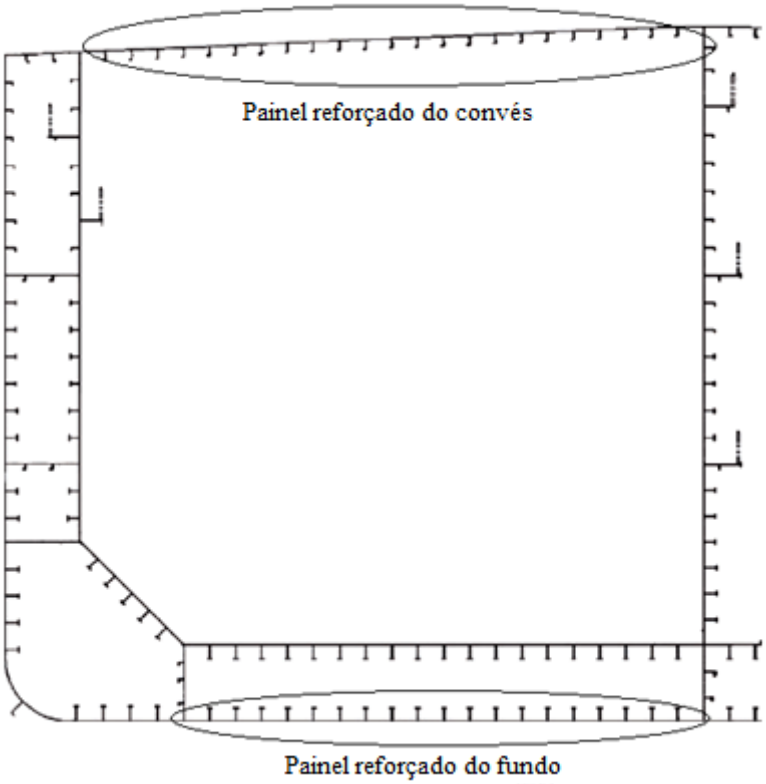


Figura 5. 1 – Corte transversal da secção mestra do navio, com representação dos painéis reforçados. Adaptado de [9]

A tabela 5.1 mostra as principais dimensões do navio.

Tabela 5. 1 – Principais dimensões do navio. Adaptado de [37]

Comprimento [m]	234
Largura [m]	42
Altura [m]	21

A estrutura do navio é feita de aço de alta resistência 32-AH, com tensão de cedência σ_c = 315 MPa, módulo de Young E = 205,8 GPa e coeficiente de Poisson v = 0,3.

As figuras 5.2 e 5.3 representam os painéis reforçados do convés e do fundo do navio, respetivamente, sendo apresentadas as espessuras de cada painel ao longo do seu comprimento.

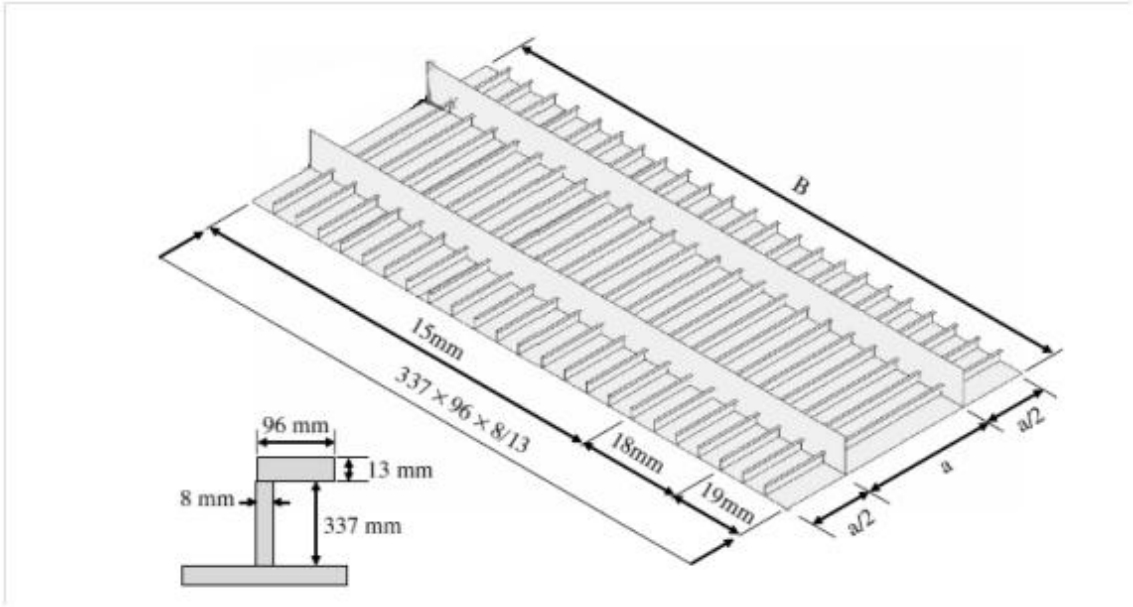


Figura 5. 2 – Painel reforçado do convés [9]

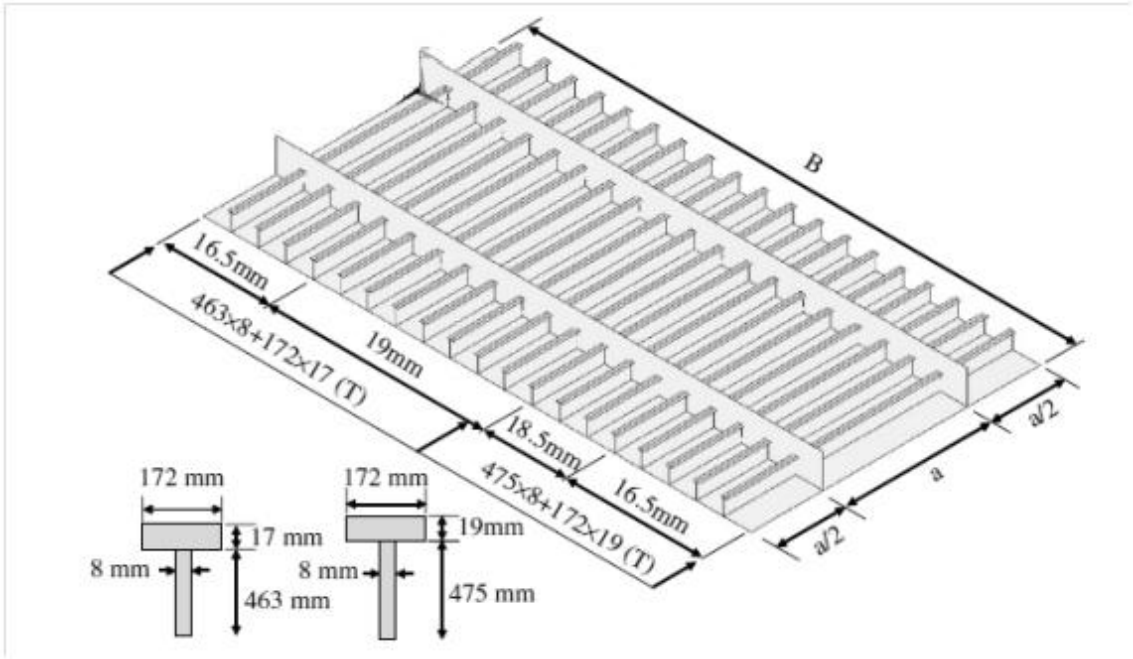


Figura 5. 3 – Painel reforçado do fundo [9]

A figura 5.4 representa o painel reforçado do fundo, modificado, que é utilizado no presente exemplo e no qual se considera constante a espessura do fundo. Na figura 5.5 representa-se o mesmo painel, mas destacando um conjunto de placas idênticas à estudada neste capítulo.

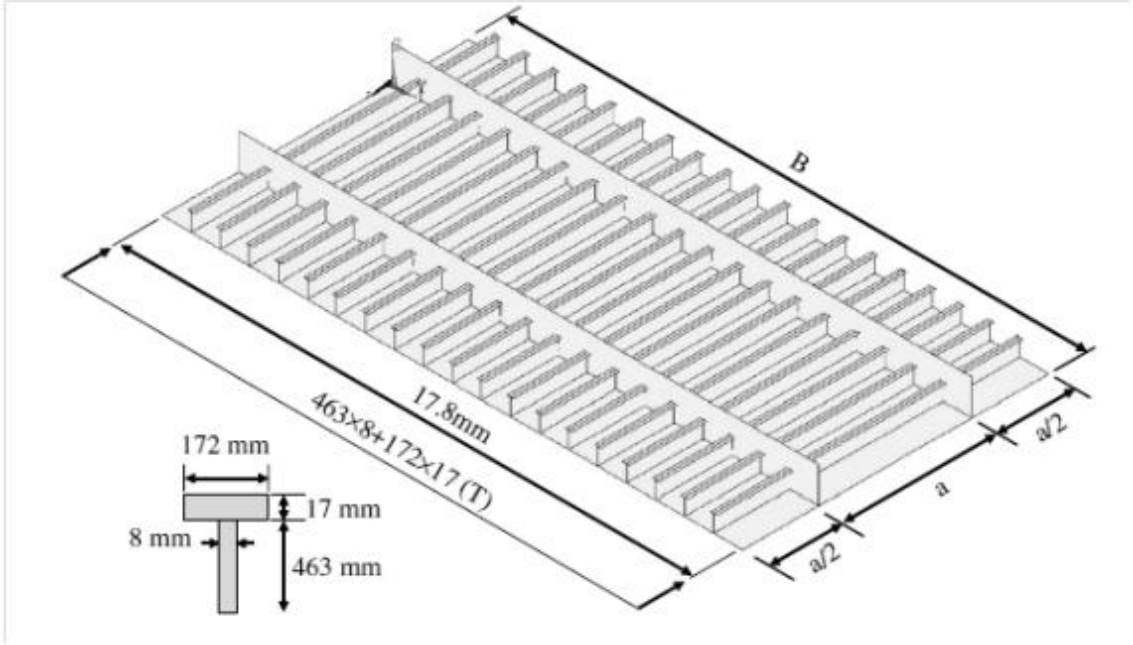


Figura 5. 4 – Painel reforçado do fundo (Modificado) [9]

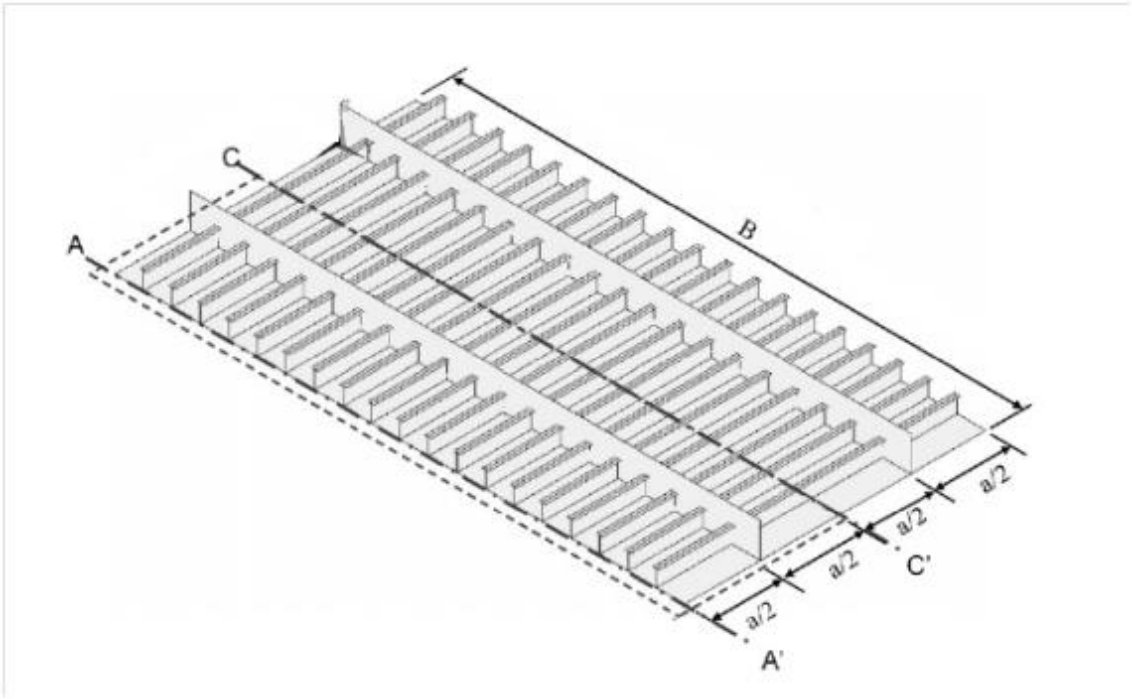


Figura 5. 5 – Painel reforçado do fundo (Modificado), com destaque de um conjunto de placas [9]

Considerando apenas um carregamento constituído por tensões normais resultantes dos momentos de alquebramento (quando ocorre uma maior concentração de pesos nas extremidades da embarcação, provocando uma curvatura longitudinal com a convexidade para baixo) e de contra-alquebramento (quando ocorre uma maior concentração de pesos no centro da embarcação, provocando uma curvatura longitudinal com a curvatura para cima) verifica-se ser suficiente, devido à simetria, estudar apenas um troço de comprimento *a* e largura *b* como o representado na figura 5.6, onde *a* é a distância entre reforços transversais. O modelo estudado contém apenas um reforço longitudinal em T e é atravessado por um único reforço transversal, representado pela linha ponteada na figura 5.6.

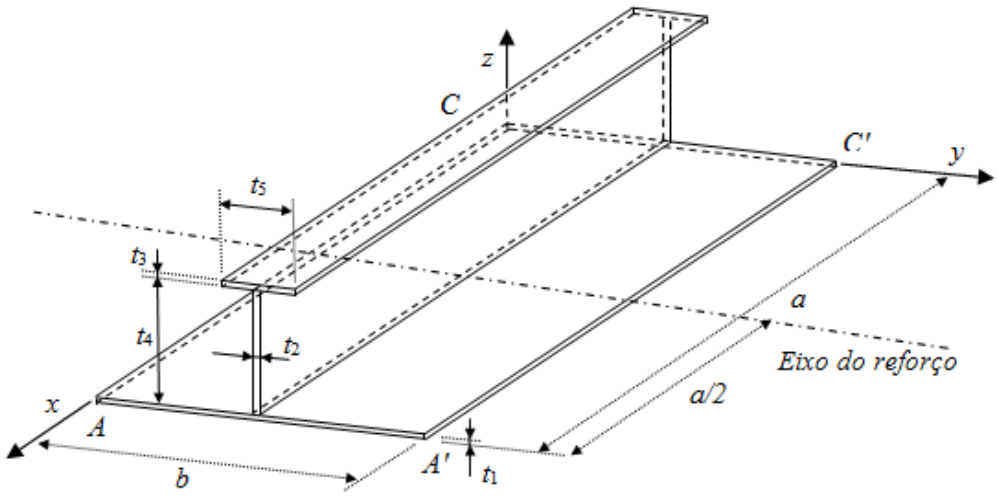


Figura 5. 6 – Modelo da placa reforçada

A tabela 5.2 contém as dimensões da placa reforçada

Tabela 5. 2 – Dimensões da placa reforçada

Variável	Unidades	Valor nominal
a	[mm]	4300,0
b	[mm]	815,0
t_1	[mm]	17,8
t_2	[mm]	8,0
t_3	[mm]	17,0
t_4	[mm]	463,0
t_5	[mm]	172,0

Por forma a realizar as análises não lineares de elementos finitos para obter os conjuntos de treino e de simulação para a rede neuronal que se pretende criar, utilizou-se um algoritmo de MATLAB desenvolvido por Araújo [36], no qual as seguintes condições de fronteira foram aplicadas. *T*[*x, y, z*] indica as restrições de translação e *R*[*x, y, z*] as restrições rotacionais segundo as coordenadas *x*, *y* e *z* respetivamente. Um "0" indica que a coordenada apresenta restrição enquanto um "1" indica que não há restrição. (1 - livre , 0 - fixo)
Fronteira *AC* e *A'C'* (condições de simetria) = *T*[1, 0, 1], *R*[0, 1, 1]
Fronteira *AA'* (condições de simetria) = *T*[*Ux*, 1, 1], *R*[1, 0, 0]
Fronteira *CC'* (condições de simetria) = *T*[0, 1, 1], *R*[1, 0, 0]
Nos pontos ligados ao reforço transversal (floor) = *T*[1, 1, 0], *R*[1, 1, 1] na placa e *T*[1, 0, 1], *R*[1, 1, 1] na alma do reforço.

Também se chama a atenção para o facto de em [9] se ter estudado um painel com vários reforços transversais pois se considerou um carregamento constituído por tensões normais segundo *x* e *y* e ainda a pressão da água no fundo. Por isso as condições fronteira na aresta *AC* e *A'C'* são diferentes no modelo estudado e no artigo.

Estas condições fronteira são iguais às indicadas em [36] e [38].

Apenas diferem das apresentadas em [9] na fronteira *AC* e *A'C'* onde este último autor indica [1, 1, 0, 1, 0, 0], e cada aresta tem deslocamento idêntico segundo *y*. Foram feitas análise com as condições em *AC* e *A'C'* admitidas em [9] tendo-se concluído que não afetavam significativamente a tensão de colapso.

Tal como indicado em [9] são consideradas deformações iniciais favorecendo o colapso por instabilidade da placa reforçada que simulam as imperfeições da geometria que ocorrem nos painéis construídos por soldadura. A deformação inicial considerada é a sobreposição de 3 diferentes deformações, representadas nas figuras 5.7 a 5.9.

1. Deformação inicial da placa segundo *z* de amplitude w_p e forma aproximada ao modo de instabilidade da placa (*m*=5)

$$w_p = \frac{b}{200} \tag{5.1}$$

$$z_1 = w_p \cdot \cos\left(\frac{m \cdot \pi \cdot x}{a}\right) \cdot \cos\left(\frac{\pi \cdot y}{b}\right) \tag{5.2}$$

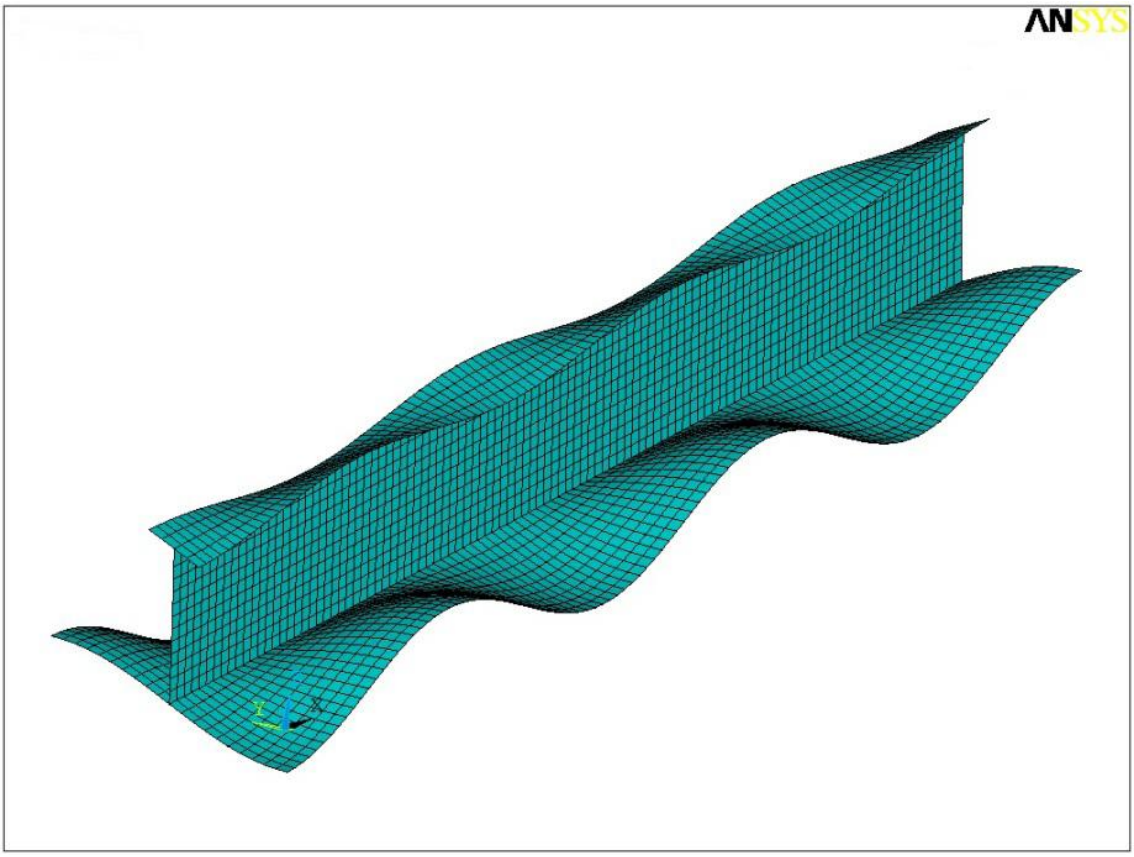


Figura 5. 7 - Deformação inicial da placa (fator amplificação de 30x) [36]

2. Deformação inicial com a forma aproximada ao modo de instabilidade da viga-coluna, e com amplitude segundo *z* igual a:

$$w_{oc} = \frac{a}{1000} \tag{5.3}$$

$$z_2 = w_{oc} \cdot \cos\left(\frac{\pi \cdot x}{a}\right) \tag{5.4}$$

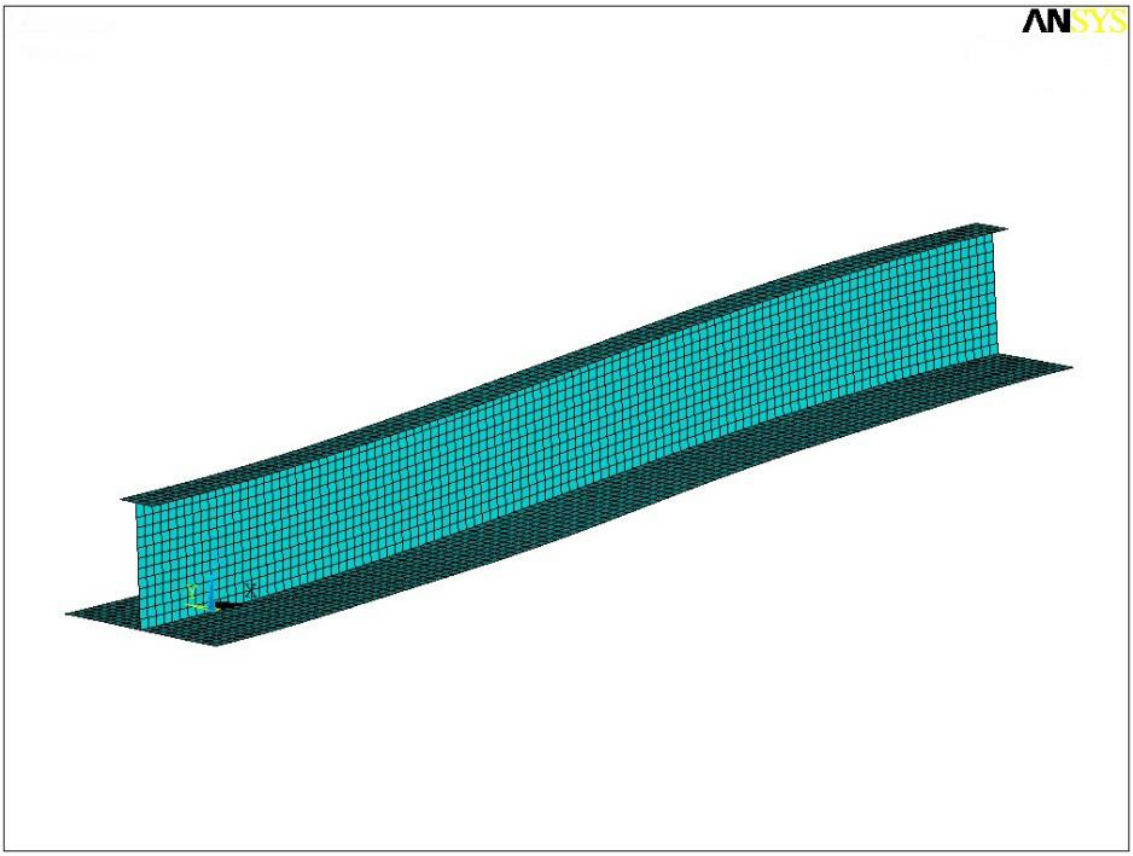


Figura 5. 8 - Deformação inicial do modo de instabilidade da viga-coluna (fator amplificação de 30x) [36]

3. Deslocamento lateral da alma do reforço com a forma aproximada ao modo de instabilidade da viga-coluna, e com uma amplitude *y* igual a:

$$w_s = \frac{a}{1000} \tag{5.5}$$

A posição da alma segundo z varia com a função seno, assumindo o valor máximo no banzo. O deslocamento lateral é nulo nos nós que estão ligados ao reforço transversal, isto é, para *x = a/2*

$$y_1 = w_{os} \cdot \text{sen}\left(\frac{\pi \cdot z}{2 \cdot h_w}\right) \cdot \cos\left(\frac{\pi \cdot x}{a}\right) \tag{5.6}$$

O valor $z_1 + z_2$ deve ser somado a todos os nós da malha. O valor de y_1 deve ser somado à coordenada *y* dos nós da alma e do banzo do perfil T.

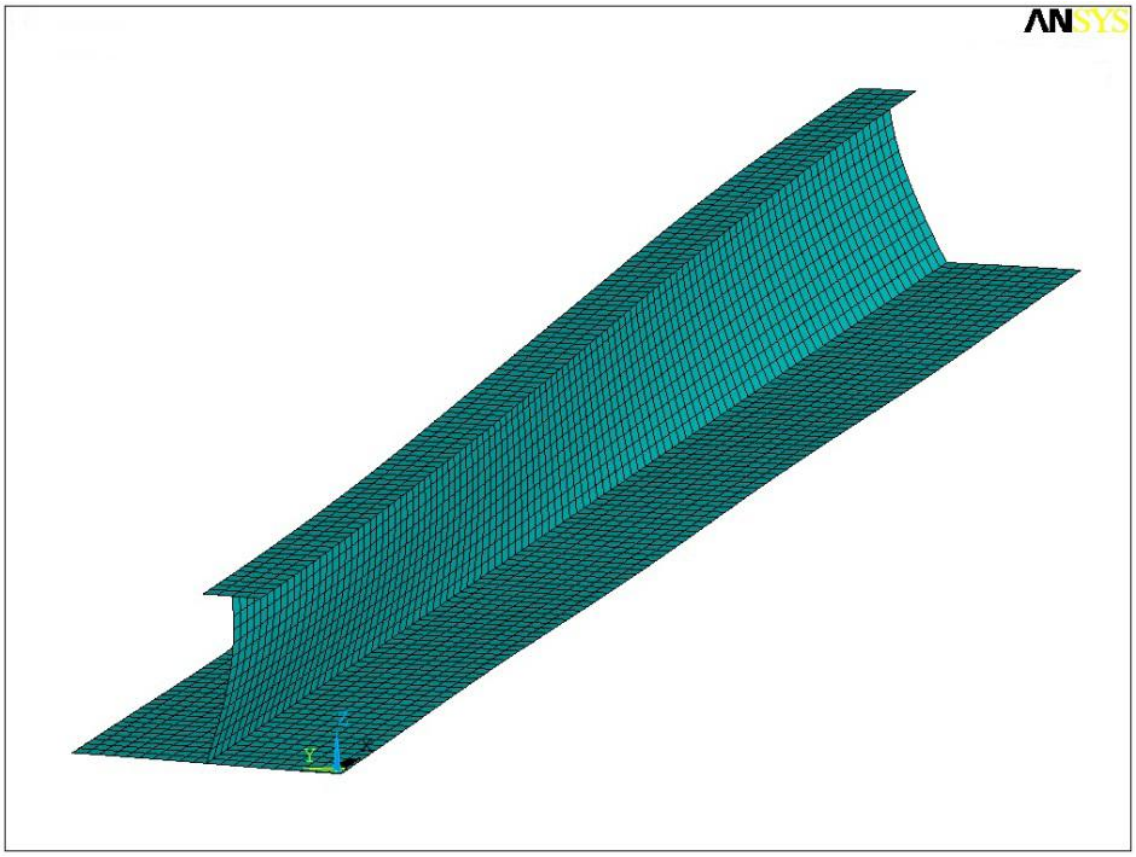


Figura 5. 9 - Deformação inicial da alma do reforço (fator amplificação de 30x) [36]

Em [9] não é apresentada qualquer análise de fiabilidade do modelo da placa reforçada. Para efetuar análises de fiabilidade que permitam testar a metodologia desenvolvida nesta dissertação seguem-se as opções indicadas em [10].

A carga axial exercida sobre o painel é obtida assumindo distribuições probabilísticas para o momento fletor induzido pelas ondas, M_{wv} , e para o momento fletor em águas paradas, M_{sw} . Consideraram-se os valores mencionados em [10], com as unidades em MN·m.

Variáveis envolvidas no cálculo da ação

Tabela 5. 3 – Variáveis que condicionam as ações sobre a placa reforçada

Designação	Variável	Distribuição	Média, μ	Desvio, σ
M_{sw}	Momento águas paradas	Normal	1483,7	430,3

$\boldsymbol{M_{wv}}$	Momento induzido ondulação	Gumbel	4603,153	401,181

Para calcular a média e o desvio padrão da distribuição de M_{wv} consideram-se os parâmetros μ (localização) = 4422,6 e β (escala) = 312,8 como indicado em [10].

Assume-se que a força axial pode ser calculada a partir dos valores dos dois momentos fletores adotando para o módulo resistente da secção transversal um valor determinístico. Consideraram-se dois valores possíveis, Z = 41,1 m^3 e Z = 33,7 m^3.

A tensão atuante na placa reforçada é calculada a partir dos valores dos momentos, pela expressão:

$$f_a = \frac{M_{sw} + M_{wv}}{Z} \tag{5.7}$$

Para calcular a resistência do painel consideraram-se 8 variáveis aleatórias e todos os restantes parâmetros foram considerados determinísticos. O modelo estocástico para as variáveis foi obtido assumindo um tipo de distribuição e um valor do coeficiente de variação, ilustrado na tabela 5.4.

Em seguida, considerando que o valor inicial ou de projeto apresentado em [9] correspondia a um determinado percentil, obtiveram-se os parâmetros da distribuição, como se mostra na tabela 5.5.

Variáveis envolvidas no cálculo da resistência

Tabela 5. 4 – Variáveis que condicionam a resistência

Designação	Valor de projeto	Distribuição	Valor de projeto corresponde ao percentil	COV
t_1	17,8 mm	Normal	0,50	0,02
t_2	8,0 mm	Normal	0,50	0,02
t_3	17,0 mm	Normal	0,50	0,02
E	205,8 GPa	Lognormal	0,50	0,10
σ_c	315 MPa	Lognormal	0,05	0,08
w_p	4,075 mm	Lognormal	0,95	0,50

w_{oc}	4,300 mm	Lognormal	0,95	0,50
w_{os}	4,300 mm	Lognormal	0,95	0,50

Tabela 5. 5 – Modelo probabilístico das variáveis que condicionam a resistência da placa

Designação	Variável	Distribuição	Média, μ	Desvio, σ
t_1	Espessura placa	Normal	17,8	0,356
t_2	Espessura alma reforço	Normal	8,0	0,16
t_3	Espessura banzo reforço	Normal	17,0	0,34
E	Módulo de elasticidade	Lognormal	205,8	20,58
σ_c	Tensão de cedência	Lognormal	360,37	28,83
w_p	Empeno da placa	Lognormal	2,095	1,048
w_{oc}	Empeno longitudinal	Lognormal	2,211	1,106
w_{os}	Empeno reforço	Lognormal	2,211	1,106

Algoritmo FORM

Com o modelo probabilístico de todas as variáveis envolvidas foram realizadas análises de fiabilidade para a placa reforçada com o algoritmo FORM implementado em MATLAB, tendo-se obtido os resultados da tabela 5.6:

Tabela 5. 6 – Resultados FORM

Probabilidade de colapso pelo método FORM		
Z (m^3)	β	p_c
33,7	3,60	$1{,}5915 \times 10^{-4}$
41,1	4,87	$5{,}5877 \times 10^{-7}$

Contudo devido à deficiente precisão no cálculo da tensão de colapso foi necessário usar diferenças finitas centrais com delta = 0,5 × σ (desvio padrão) o que é muito superior ao normalmente referido na literatura.

Geração dos conjuntos de treino e de teste

Uma rede neuronal aprende uma função para a qual não existe uma expressão explícita a partir unicamente dos valores da função obtidos para vários pontos do domínio. Para aplicar esta técnica ao problema da placa reforçada é necessário calcular a tensão de colapso para vários conjuntos de valores das variáveis, com recurso ao ANSYS. Com o objetivo de cobrir eficientemente o domínio de cada variável e simultaneamente manter o número de cálculos tão reduzido quanto possível usou-se a técnica de amostragem por hipercubo latino e a função *lhsdesign* do MATLAB. Foi ainda usada a opção de redução da correlação para assegurar uma perfeita cobertura do domínio, especificando na função *lhsdesign* o parâmetro *'criterion', 'correlation'*. Geraram-se dois ficheiros contendo respetivamente 1000 e 2000 conjuntos diferentes de valores das 8 variáveis aleatórias associadas à resistência assim como a tensão de colapso correspondente.

O procedimento para utilizar o hipercubo latino requer a definição do domínio para cada variável. Este é contínuo e delimitado pelos valores máximo e mínimo e, neste trabalho, foram tomados os valores que correspondiam à probabilidade acumulada de 0,005 e de 0,995. Como a função *lhsdesign* gera valores entre 0 e 1, utiliza-se uma transformação para os converter para o domínio desejado. Cada um dos ficheiros com 1000 ou 2000 ocorrências tem por isso o valor das 8 variáveis indicadas na tabela 5.5 mais a tensão de colapso calculada pelo ANSYS.

Treino da rede neuronal

Utilizando o ficheiro com 2000 conjuntos de valores gerado anteriormente procedeu-se ao treino da rede com a interface *nntool* do MATLAB.

Para isso os dados constantes dos dois ficheiros são lidos para memória e simultaneamente normalizados para que o domínio de cada variável fique compreendido entre 0 e 1. De facto a ferramenta de redes neuronais do MATLAB supostamente normaliza sozinha cada uma das variáveis. Contudo, depois de ter introduzido valores não normalizados para treino da rede, verificou-se que era impossível efetuar o treino. A *nntool* divide automaticamente o conjunto de valores de treino em 3 grupos, reservando 60% para treino, 20% para validação e 20% para teste. A validação consiste em medir o erro que a rede comete quando generaliza e permite controlar o processo do treino que termina quando este erro é suficientemente reduzido. O erro no subconjunto de teste não tem influência sobre o processo do treino e permite uma medida independente do desempenho da rede treinada. É possível visualizar o erro quadrático médio para os 3 subconjuntos durante o processo do treino de forma a avaliar o desempenho da rede treinada, como se pode ver na figura 5.10. Depois da rede treinada foi realizado um teste independente utilizando o outro conjunto com 1000 elementos.

Para cada elemento foi calculado o erro relativo e escolhido o valor máximo, ilustrado na tabela 5.7.

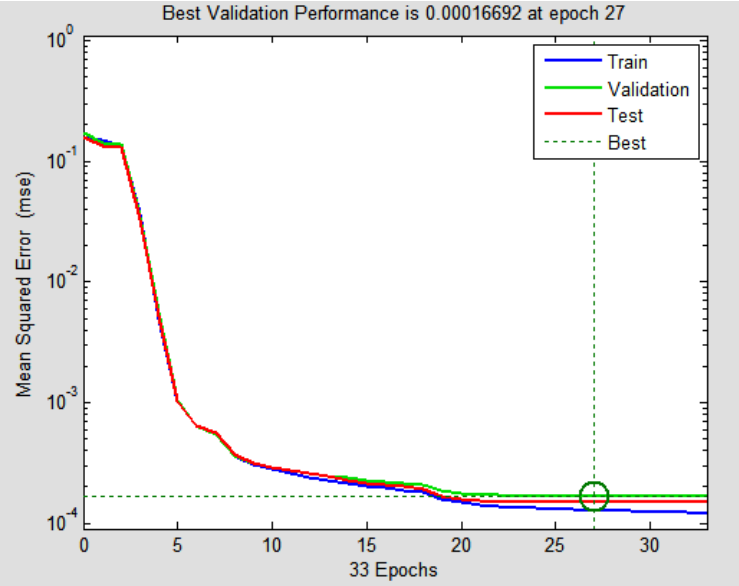


Figura 5. 10 – Erro quadrático médio para os subconjuntos de treino, validação e teste durante o processo de treino da rede neuronal

Tabela 5. 7 – Erros relativos máximos

Número de neurónios	Erro relativo máximo [%]
10	1,98
15	2,29
20	1,98
25	1,53
30	1,97
35	1,95
40	1,91

Analisando a tabela 5.7 escolheu-se a rede com 25 neurónios na camada intermédia para simular a tensão de colapso na placa reforçada

Análise de fiabilidade pelo método de Monte Carlo com redes neuronais

Finalmente foram efetuadas análises de fiabilidade pelo método de Monte Carlo usando as distribuições de todas as variáveis envolvidas. Gerando ocorrências das 8 variáveis que condicionam a resistência e usando em seguida a rede neuronal treinada obtiveram-se ocorrências da tensão de colapso que foram usadas na função de estado limite juntamente com as ocorrências dos dois momentos atuantes. Os resultados para um número de ocorrências $N = 1 \times 10^8$ são os apresentados na tabela 5.8.

Tabela 5. 8 - Resultados do método de Monte Carlo com redes neuronais

Método de Monte Carlo com rede neuronal		
Z (m^3)	β	p_c
33,7	3,98	$3{,}5178 \times 10^{-5}$
41,1	5,26	$7{,}4000 \times 10^{-7}$

Verifica-se que o número de colapsos ocorridos na simulação com Z = 41,1 m^3, 40 é demasiado reduzido para que se possa prever com precisão o valor da probabilidade de colapso. O valor obtido deve ser considerado apenas uma aproximação do valor exato.

Comparação de resultados

Tabela 5. 9 – Comparação de resultados

	β	p_c
Z = 33,7 m^3		
FORM	3,60	$1{,}5915 \times 10^{-4}$
Monte Carlo + RNA	3,98	$3{,}5178 \times 10^{-5}$
Z = 41,1 m^3		
FORM	4,87	$5{,}5877 \times 10^{-7}$
Monte Carlo + RNA	5,26	$7{,}4000 \times 10^{-7}$

Verifica-se que para Z = 33,7 m^3 os valores FORM e Monte Carlo + RNA são muito idênticos (diferenças de 0,3% para β e de 3,6% para p_c).

Para Z = 41,1 m^3 os valores FORM e Monte Carlo + RNA são menos próximos. Neste caso o valor obtido por Monte Carlo + RNA é apenas uma aproximação, pois, para o nível de probabilidade atingido, deveriam ser usadas mais amostras (10^9) para obter um valor preciso de p_c.

Capítulo6

Conclusões e Desenvolvimentos Futuros

O objetivo principal desta dissertação consistia em testar a capacidade das redes neuronais do MATLAB para aproximar a resposta de estruturas de comportamento não linear, comparando os resultados obtidos, com os obtidos com análises por elementos finitos realizadas com o programa ANSYS.

Outro objetivo foi aplicar as RNA na determinação da fiabilidade de estruturas, comparando os resultados obtidos, com os calculados por outros métodos usados nesse domínio, como o FORM.

A necessidade de aprofundar os conhecimentos sobre fiabilidade de estruturas, levou a uma pesquisa e estudo no âmbito da segurança estrutural. Foram estudados os diferentes métodos de análise de segurança estrutural, conduzindo inevitavelmente aos conceitos probabilísticos que sustentam os métodos de nível 2 e 3. Os requisitos para a segurança estrutural foram analisados tendo com base a legislação em vigor, sendo apresentados os níveis de segurança propostos pelos documentos "JCSS – Probabilistic Model Code" [2] e "Eurocódigo 0" [3]. Analisando o problema fundamental da fiabilidade estrutural, foram explicados os métodos de cálculo da probabilidade de colapso de uma estrutura, destacando-se o método de Monte Carlo utilizado nesta dissertação.

O aprofundamento de conhecimentos sobre RNA tornou-se indispensável, conduzindo a uma pesquisa e estudo sobre o tema onde se abordaram os princípios biológicos deste método

em comparação com o modelo artificial. Uma resenha histórica foi também realizada, por forma a permitir acompanhar a evolução das RNA ao longo dos tempos. Dos tipos de redes e treino destacou-se a *multilayer feedforward network*, explicando-se pormenorizadamente o algoritmo de retropropagação utilizado nesta dissertação.

Num dos exemplos estudados, tendo havido necessidade de diminuir o número de análises de elementos finitos, sem no entanto deixar de cobrir o domínio pretendido, estudou-se a técnica do hipercubo latino por forma a saber como aplica-la.

Foram também descritos os conceitos de otimização estrutural bem como de algoritmos genéticos, também utilizados nesta dissertação.

Por forma a testar a capacidade das redes neuronais do MATLAB utilizou-se uma função de teste de comportamento não linear, a qual serviu para ilustrar o funcionamento das interfaces gráficas de redes neuronais artificiais do MATLAB. Os resultados obtidos foram bastante satisfatórios, dando assim segurança para prosseguir o trabalho e aplicar as RNA para aproximar o comportamento de estruturas.

A primeira estrutura estudada foi uma treliça de 6 barras, simplesmente apoiada, na qual se pretendeu fazer otimização da massa da estrutura, com constrangimentos de fiabilidade. A otimização foi feita considerando que o colapso da estrutura ocorra, quando pelo menos uma das barras ultrapassar a tensão de cedência. Ou seja, trata-se de um sistema em série, no qual a falha de um dos elementos, implica uma falha no sistema. Desta forma foram consideradas seis funções de falha, uma para cada barra. Assumiu-se que a probabilidade de colapso p_c, não deveria exceder $p_c = 0{,}001$, que corresponde a um índice de fiabilidade $\beta = 3{,}090$, tal como em [7] e [35]. Para o cálculo da probabilidade de colapso do sistema, recorreu-se ao método de Monte Carlo associado à rede neuronal.

Recorreu-se ao programa de elementos finitos ANSYS por forma a obter valores das tensões das barras para os conjuntos de treino e de teste. Utilizando a interface gráfica do MATLAB, *nftool*, para treinar algumas redes neuronais, com diferente número de neurónios na camada intermédia, escolheu-se depois a mais eficiente, sendo a utilizada para gerar valores das tensões por forma a calcular a probabilidade de colapso através do método de Monte Carlo.

Recorrendo à interface gráfica de algoritmos genéticos do MATLAB, realizou-se a otimização com constrangimentos de fiabilidade, através de uma função que calcula o mérito para cada cromossoma, penalizando a massa da treliça sempre que a probabilidade de colapso pretendida é excedida.

Os resultados obtidos foram bastante satisfatórios em comparação com os obtidos em [7] e [35].

Por fim foi analisada uma placa reforçada da estrutura do casco de um navio. Após explicar a localização da placa reforçada no navio onde está inserida, foram feitas as simplificações consideradas necessárias. Foram também impostos empenos iniciais na placa bem como condições de fronteira.

Um algoritmo em linguagem MATLAB foi utilizado para criar o código APDL que permitiu as análises por elementos finitos realizadas pelo ANSYS. Por forma a cobrir eficientemente o domínio de cada variável e simultaneamente manter o número de cálculos tão reduzido quanto possível usou-se a técnica de amostragem por hipercubo latino e a função *lhsdesign* do MATLAB. Foi ainda usada a opção de redução da correlação para assegurar uma perfeita cobertura do domínio. Dois ficheiros contendo respetivamente 1000 e 2000 conjuntos diferentes de valores das 8 variáveis aleatórias associadas à resistência assim como a tensão de colapso correspondente, foram gerados e utilizados respetivamente para simulação e treino da rede neuronal.

Com o modelo probabilístico de todas as variáveis envolvidas foram realizadas análises de fiabilidade para a placa reforçada com o algoritmo FORM implementado em MATLAB e pelo método de Monte Carlo em conjunto com a rede neuronal, sendo os resultados obtidos apresentados na tabela 5.9.

Dada a semelhança entre os resultados obtidos pelos dois métodos, considera-se que os resultados são coerentes e a metodologia utilizada bastante eficiente.

Verificou-se, através dos exemplos estudados que a ferramenta de redes neuronais do MATLAB tem a capacidade de aproximar a resposta de estruturas de comportamento não linear. Com efeito, comparando os resultados obtidos com os calculados pelo ANSYS constatou-se que os erros eram sempre de pequena amplitude.

Essa capacidade das RNA de aproximar a resposta não linear de estruturas foi usada com sucesso em analise de fiabilidade, através da combinação do método de Monte Carlo com as RNA. Com efeito o método de Monte Carlo é de fácil utilização e permite calcular a probabilidade de colapso de estruturas com várias funções de falha e com várias variáveis aleatórias apresentando como maior desvantagem o tempo de cálculo requerido quando a probabilidade de falha é muito reduzida. A utilização em conjunto com as RNA elimina esse inconveniente.

Os resultados obtidos são comparados com os publicados na literatura ou com valores obtidos usando outros métodos de análise de fiabilidade, tendo-se constatado que eram coerentes.

Desta forma dão-se como plenamente atingidos os objetivos desta dissertação.

Desenvolvimentos Futuros

Dado que a investigação realizada no âmbito desta dissertação se insere no projeto PTDC/ECM/115932/2009 – "Métodos adaptativos para a análise de fiabilidade de estruturas complexas", são vários os desenvolvimentos futuros para continuar o presente trabalho.

Existindo algumas fases morosas neste processo, nomeadamente na geração dos conjuntos de treino e de teste da rede neuronal, apresenta-se a paralelização do cálculo das análises de elementos como uma solução para reduzir o tempo de computação.

A compilação dos algoritmos desenvolvidos em linguagem MATLAB ou a sua transcrição para a linguagem C++ é também uma possibilidade que permitiria reduzir o tempo de cálculo.

Bibliografia

[1] Melchers, R. E., “Structural reliability analysis and prediction”, Wiley Editorial, 2nd edition, 1999;

[2] Joint Committee on Structural Safety, "Probabilistic Model Code - Part 1: Basis of Design", 2000;

[3] EN 1990, European Committee for Standardization, "Eurocode 0 - Basis of Structural Design", 2001;

[4] Zienkiewicz, O. C.; Taylor, R. L.; Zhu, J. Z.; “The finite element method: its basis and fundamentals”, Butterworth – Heinemann, 6th edition, 2005;

[5] Gonçalves, C. M. M., “Estudo da Integridade da Secção-Mestra das Lanchas de Fiscalização Rápida da Classe “Centauro” ”, Dissertação de Mestrado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2011;

[6] Amaral , Alexandre Bee; Barbosa, Gabriel Grando; Netto, Hugo Beguetto; Camargo, Jairo Henrique Melara de; Filho, Marcos Antonio Costantin; Martinez, Maria Angelica Castelli; Dutra, Paola; Nunes, Paulo Afonso; Machado, Ralph Magalhaes; Chrestenzen; Pieralisi, Ricardo; Dalmagro, Taiane; Matos, Thamires da Silva; Kramer, Vanessa Cristina; “ANSYS 9.0 – Análises Estruturais”; PET Engenharia Civil – Universidade Federal do Paraná; 2010;

[7] Cardoso, J. B.; Almeida, J. R.; Dias, J. M.; Coelho, P. G.; ” Structural reliability analysis using Monte Carlo simulation and neural networks”, Adv Eng Softw, 2008;

[8] Roseiro, Luís Manuel Ferreira, “Detecção de dano e identificação estrutural com redes neuronais artificiais”, Tese de Doutoramento, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 2004;

[9] Paik, J. K.; Kim, B. J.; Seo, J. K.; “Methods for ultimate limit state assessment of ships and ship-shaped offshore structures: Part II—stiffened plates”, Ocean Engineering, 2008;

[10] Gaspar, B.; Naess, A.; Leira, B. J.; Soares C. G.; “Efficient system reliability analysis by finite element models”, OMAE, 2011;

[11] Gonçalves, Rui Carlos Rodrigues Fernandes, “Robustez de estruturas de betão armado sujeitas a deterioração”, Dissertação de Mestrado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2011;

[12] Amorim, Miguem Ângelo Guerra, "Análise Estocástica da Robustez de Estruturas Porticadas em Madeira ", Dissertação de Mestrado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2010;

[13] Figueiredo, Joana Maria de Oliveira, "Análise Probabilística da Robustez de Estruturas de Madeira ", Dissertação de Mestrado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2011;

[14] Guerreiro, Luís, "Introdução à Fiabilidade Estrutural – Apontamentos da Disciplina de Dinâmica e Engenharia Sísmica ", Instituto Superior Técnico, 1999;

[15] Lastiti, Cayo Pardos e Pauletti,Ruy Marcelo de Oliveira, "Aplicação de redes neuronais artificiais à engenharia de estruturas", 2004;

[16] Martins, Fernando Gomes, "Aplicação de Redes Neuronais Artificiais em Simulação e Controlo de Processos Químicos", Faculdade de Engenharia da Universidade do Porto, 1997;

[17] McCulloch, W.; Pitts, W.; "A logical calculus of the ideas of samples needed activity", Bulletin of mathematical biophysics, 1943;

[18] Rosenblatt, F.; "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain", Psychological Review, 1958;

[19] Widrow, B.; Hoff, M.; "Adaptive Switching Circuits", IRE WESCON Convention Record, 1960;

[20] Minsky, M.; Papert, S.; "Perceptrons", MIT Press, Cambridge, USA, 1969;

[21] Hopfield, J.; "Neural Networks and Physical Systems with Emergent Collective Computational Propertie", Proc. Nat. Academy of Sciences, USA, 1982;

[22] Rumelhart, D.; Hinton, G.; Williams, R.; "Learning Internal Representations By Error Propagation", PDP Research Group, Parallel Distributed Processing, MIT Cambridge, USA, 1986;

[23] Barreto, Jorge M.; "Introdução às Redes Neurais Artificiais", Laboratório de Conexionismo e Ciências Cognitivas - UFSC -Departamento de Informática e de Estatística 88040-900 - Florianópolis – SC, 2002;

[24] Cybenko, G.; "Approximation by Superposition of Sigmoidal Function", Mathematics of Control, Signal and Systems, 1989;

[25] Funahashi, K.; "On the Approximate Realization of Continuous Mappings by Neural Networks", Neural Networks, 1989;

[26] Hornik, K.; Stinchcombe, M.; White, H.; "Universal Approximation of An Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks", Neural Networks, 1990;

[27] Hornik, K.; "Aproximation Capabilities of Multilayer Feedforward Networks", Neural Networks, 1991;

[28] Lopes, Patrícia da Silva, "Detecção de danos em estruturas por meio de técnicas de redes neurais artificiais e algoritmos genéticos", Dissertação de Mestrado, Universidade Federal de Itajubá – Instituto de Engenharia Mecânica, 2007;

[29] Estêvão, J.M.C., Jesus, Mário, "Aplicação de Algoritmos Genéticos na Optimização de Secções de Vigas de Betão Armado", Universidade do Algarve, 2008;

[30] Olson A., Sandberg G., Dahlblom O., "On Latin hypercube sampling for structural reliability analysis", Structural Safety, 2002;

[31] Maschio C, Carvalho C.P.V., Schiozer D.J., "Aplicação da técnica do hipercubo latino na integração do ajuste de histórico com a análise de incertezas", Departamento de Engenharia de Petróleo – Faculdade de Engenharia Mecânica – Universidade Estatual de Campinas, 2009;

[32] Cordeiro, Marcelo de França, "Uma Técnica para Otimização Estrutural Mediante a Derivada Topológica", 2007;

[33] Ortigueira, Manuel Duarte, "Manual de Introdução ao MatLab", Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2002;

[34] Beagle, Mark Hudson; Hagan, Martin T.; Demuth, Howard B.; "Neural Network Toolbox™ 7 User's Guide", MathWorks, 2010;

[35] Burton, S. A.; Hajela, P.; "A variable-complexity approach to secondorder reliability-based optimization", Struct Multidiscip Optimization, 2003;

[36] Araújo, Miguel Melvill; "Estudo comparativo de painéis reforçados em liga de alumínio ou em aço de construção naval", Dissertação de Mestrado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2012;

[37] Joint Tanker Project, "Common ctructural rules for double hull oil tankers", 2005;

[38] M. Corak, J. Parunov, A.P. Teixeira e C. Guedes Soares, "Performance of the common structural rules design formulations for the ultimate strength of uniaxially loaded plates and stiffened panels".