Zenon Waszczyszyn
Editor

International Centre
for Mechanical Sciences

Advances of Soft Computing in Engineering

CISM Courses and Lectures, vol. 512

SpringerWienNewYork

SpringerWienNewYork

CISM COURSES AND LECTURES

The series presents lecture notes, monographs, edited works and proceedings in the field of Mechanics, Engineering, Computer Science and Applied Mathematics.
Purpose of the series is to make known in the international scientific and technical community results obtained in some of the activities organized by CISM, the International Centre for Mechanical Sciences.

INTERNATIONAL CENTRE FOR MECHANICAL SCIENCES

COURSES AND LECTURES - No. 512

ADVANCES OF SOFT COMPUTING IN ENGINEERING

EDITED BY

ZENON WASZCZYSZYN
RZESZOW AND CRACOW UNIVERSITIES
OF TECHNOLOGY, POLAND

SpringerWienNewYork

This volume contains 215 illustrations

Printed in Italy
SPIN 12775583

All contributions have been typeset by the authors.

ISBN 978-3-211-99767-3 SpringerWienNewYork

PREFACE

An increasing interest in the neural networks and soft computing is visible in sciences and engineering. Just in this field, two CISM Advanced Schools were organized in 1998 and 2003. The corresponding books were published as CISM Courses and Lectures Nos 404 and 496. The first book was written on neural networks in the analysis and design of structures. Chapter 7 of the other book was devoted to applications of neural networks to the identification of structural mechanics problems.

The present book corresponds to six cycles of lectures given at the CISM Advanced School on Advances of Soft Computing in Engineering, held in Udine, Italy on October 8-12, 2007. The lectures were delivered by invited professors from six different universities.

The first three Chapters are based on soft methods related to genetic and evolutionary algorithms. Next to the theoretical and algorithmic background, many engineering applications are discussed and, especially, those addressed to civil and mechanical engineering are worth emphasizing. The next three Chapters are devoted to neural networks (NNs) and their engineering applications. Beside the standard, deterministic NNs also probabilistic and, especially, Baysian NNs are discussed. Their applications in mechanics of structures and materials are presented from the viewpoint of civil, seismic and mechanical engineering problems.

The organizers of the School and editors of this book wish to express they cordial thanks to the invited lecturers (Professors Tadeusz Burczy□ski of Silesian University of Technology, Poland, Jamshid Ghaboussi of University of Illinois at Urbana-Champain, USA, Manolis Papadrakakis of National University of Athens, Greece, John Miles of Cardiff University, UK, Vassili Toropov of University of Leeds, UK) for their effort at delivering lectures and preparing the camera ready manuscripts. We would like to thank very much Professor Giulio Maier, Rector of CISM, for his enthusiasm, help and keen support in the organization of the School and to Professor Paolo Serafini for his constant help in the editorial work.

Zenon Waszczyszyn
Marek S□o□ski

CONTENTS

PREFACE

CHAPTER 1
Genetic algorithms for design
by J. Miles.....1

CHAPTER 2
Evolutionary and immune computations in optimal design and inverse problems
by T. Burczy□ski……57

CHAPTER 3
Applications of GA and GP to industrial optimization problems and inverse problems
by V.V. Toropov, L.F. Alvarez and O.M. Querin……133

CHAPTER 4
Advances in neural networks in computational mechanics and engineering
by J. Ghaboussi……191

CHAPTER 5
Selected problems of artificial neural networks development
by Z. Waszczyszyn and M. S□o□ski……...237

CHAPTER 6
Neural networks: some successful applications in computational mechanics
by M. Papadrakakis, N.D. Lagaros ans M. Fragiadakis…….317

CHAPTER 1

Genetic Algorithms for Design

John Miles

Cardiff School of Engineering
Cardiff University, UK

Abstract. The chapter covers two main areas, these being an introduction to the technology and techniques associated with genetic algorithms and then the second part looks at how genetic algorithms can be used to search for good topological solutions to engineering design challenges. The start of the chapter places genetic algorithms in context compared to other evolutionary algorithms and also describes the reasons why genetic algorithms are potentially useful. This is then followed by a look at the concept of a search space. Section two looks at the canonical genetic algorithm as a basic introduction to the technology and includes an examination of the main techniques used to encode the genome, fitness functions, operators and selection. Section three looks at how genetic algorithms can be used for design and chooses the specific example of the conceptual design of commercial office buildings. Section four introduces the basic concepts of topological search and explains how having the right form of representation is vital before looking at example relating to structural components and the design of domes using a genetic algorithm linked to computational geometry techniques. The final section then looks at further methods using generative representations and generative geometries as possible solutions to the need to develop powerful forms of representation for handling topological search in genetic algorithms.

1 Introduction

Genetic Algorithms belong to a group of techniques which are generally described by the collective term evolutionary computation. The other techniques within this group include genetic programming, evolutionary strategies, evolutionary programming and particle swarm analysis. The defining features of this group of algorithms are:

- Their usage of a stochastic search process employing a population of solutions rather than one point at a time;

- Their requirement for relatively little information about the nature of the problem being solved;
- Their ability to avoid premature convergence on local optima;
- Their ability to cope with constraints;
- Their ability to cope with problems involving many objectives.

In general, genetic algorithms are robust and applicable to a wide range of problems, although one must always bear in mind the findings of Wolpert and MacReady (1997) that there is no single algorithm that will perform well on all problems.

Generally, genetic algorithms are thought of as an optimization technique but this is somewhat misleading. Although they are generally an excellent method for finding good solutions which are close to the optimum, they often fail to find the actual optimum. For most problems, especially in Engineering, getting close to the optimum is sufficient and the performance of genetic algorithms is such that they generally outperform other algorithms.

If it is desirable to find the optimum, rather than an answer which is very close to it, then a successful method is to use a genetic algorithm to get close to the optimum and then a technique such as hill climbing to search around the solution found by the genetic algorithm to find the desired solution.

There is another useful feature of genetic algorithms which is rarely used but which is very powerful. This is their ability to explore a search space (i.e. the space of all possible solutions) rather than look for a single "'best"' solution. For many problems, such as design, rather than locating the "best" solution, the user can find it useful to learn about the range of possibilities and also to vary the objective function (i.e. the criteria being used to search for the solution) as more is learned about the nature of the problem. In such circumstances, a single solution is undesirable and indeed, there is rarely a "best" solution to multi-objective problems because of the trade offs between the various objectives. Genetic algorithms are an excellent technique for helping designers to find areas within the problem space that contain good solutions and additionally, the interaction between the designer and the algorithm can be highly beneficial (Parmee, 2001).

The concept of a "search space" is one that will occur throughout these notes. A search space can be defined as the entire range of possible solutions to a problem. A potential search space is shown below in Figure 1. As can be seen the space has 3 dimensions. The two horizontal dimensions will typically represent the two variables that are present in the problem (for example, for an engineering problem these could be weight and cost) and the vertical dimension then represents the "fitness" of the solution (i.e. how good the solution is). Note that the search space in this example is continuous (this is not always the case) and has many peaks of similar height. The problem therefore is, typically, to look around the search

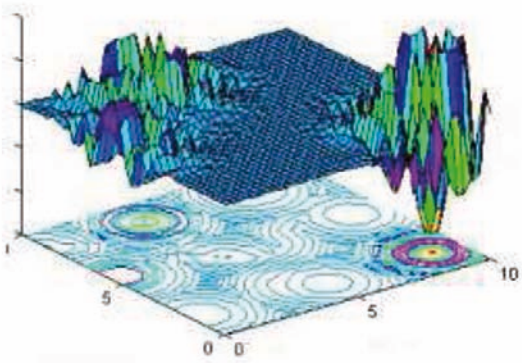


Figure 1. An example of a search space

space and find the highest peak, although sometimes it may be desirable to find all the peaks over a given threshold value. For a simple two variable problem of the sort illustrated, this looks like an easy task, but most real search spaces are multi-dimensional and many have discrete variables and so possess discontinuities which are a challenge for any algorithm.

In this chapter, the basic techniques of the so called canonical genetic algorithm will be covered. There are many variants on the canonical genetic algorithm but these are mostly fairly simple to understand once one has grasped the basic features of genetic algorithms. In terms of programming, commercial software for genetic algorithms is available. For example, Matlab has an extension which allows the easy development of a genetic algorithm. However, as will be shown in the following sections, genetic algorithms are relatively simple and often it is easier to write your own program because then this can be adapted, as required, to your particular problem.

2 The Canonical Genetic Algorithm

The basic architecture of the canonical genetic algorithm is given in Figure 2. As shown, the process starts with the creation of an initial population. Each member of the population represents a potential solution to the problem being considered although, as will be shown, many features of the problem are contained within the fitness function rather than within the population. Searching for a solution(s) using a population means that at each step, a genetic algorithm samples as many points within the search space as there are members of the population (assuming no two members are identical). This is one of the strengths of a genetic algorithm, enabling it to sample widely throughout the search space and identify areas of high performance (i.e. good solutions) on which the search can start to converge. The

use of a population enables multiple high performance areas to be identified and explored in further detail. This helps the genetic algorithm to avoid convergence on local optima.

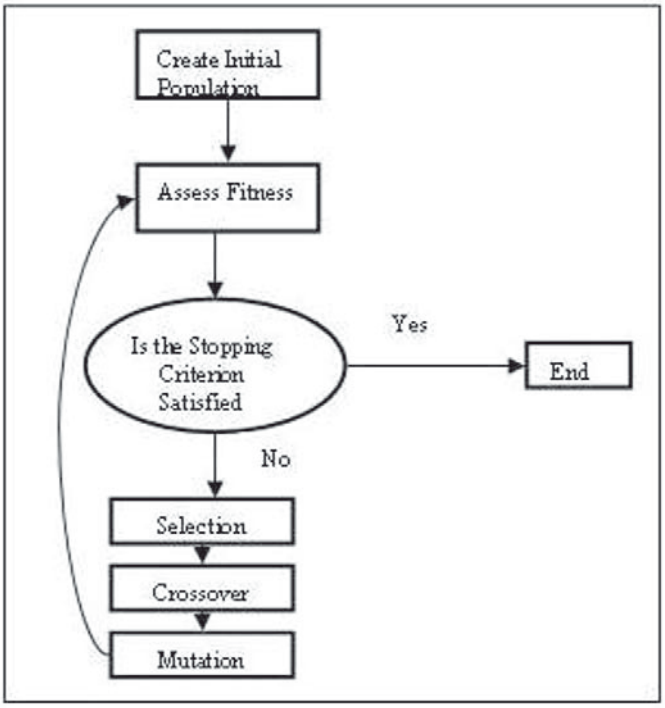


Figure 2. Schematic representation of a GA

Once the initial population is established, the basic process of the genetic algorithm is to adapt and modify the members of the population based on feedback relating to how good a solution is each member of the population, until one or more good solutions are found. The judgement of how well each member of the population performs is undertaken by the fitness function. The adaptation and modification is then undertaken by the selection, crossover and mutation processes shown in 2 and there is an iterative procedure, represented by the loop, which continues until some convergence criterion is satisfied.

The process is analogous to Darwinian evolution in that there is a population of solutions. These solutions are subject to an environment (the fitness function) which tends to favour the reproduction of the solutions which are best suited to that environment. Hence solutions which suit the defined environment are evolved over a number of iterations (called generations).

2.1 Encoding the Problem

The typical genetic algorithm uses a binary string to represent each member of the population. How this is achieved, varies with each problem. A simple practical example is the problem examined by (Hooper, 1993) who devised a technique for determining the best strategy for the disposal of the sludge from a sewage treatment works. If the sludge was to be disposed of to agriculture, then there were a given number of farms which could be used within an economical travelling distance of the treatment works. The encoding used for this problem was to represent each farm by one of the characters in a binary string (often called a chromosome), with each character being called a gene. If the gene was 1, then the policy represented by that individual was that the farm would be used to dispose of sludge. If it was zero, then the farm would not be used. Assuming an example where there are ten farms, a possible individual would then be as follows: [1001011101]

This represents the disposal of sludge on farms one, four, six, seven, eight and ten and no disposal on the remainder. In the way that a genetic algorithm works, it must be remembered that this would be just one possible solution within a population of solutions.

An alternative to using binary encoding is to use so called real numbers. The terminology is confusing because often the numbers are integers but as this is the standard terminology within the evolutionary computation community, it will be used throughout these notes.

In real number encoding, rather than using binary, actual numbers are used. For example, Bradshaw (1996) was asked to derive an optimum electricity generation strategy for the island of Great Britain. The constraints were that the strategy had to meet the demand for power at all times while allowing sufficient time for maintenance etc and taking account of the fact that it is better to generate electricity as close to the demand as possible to reduce losses in transmission. The objective of the optimisation was to reduce the damage done by the deposition of acidic gases such as Sulphur Dioxide. In this case the encoding used was a real number representation where each generating station was represented by a number between zero and 100. If for example the number was 20, then twenty per cent of that station's maximum generating capacity would be used within a given year.

Assuming an example where there are eight generating stations (in the real example there were over forty), then an example of an individual within a population would be: [35,72,80,41,0,66,7,29]

With thirty five per cent of station one's capacity being used, seventy two per cent of station two's, etc.

2.2 The Choice of Encoding

There are various arguments for and against using binary and real number encoding. The argument in favour of binary encoding is based upon the schema theory (Holland, 1975), Goldberg (1989).

The basis of the schema theory is that a schema is a similarity template that defines similarities between individuals in a population (often called chromosomes), at various points within the individuals. For example consider the following example from Parmee (2001) which uses three individuals as follows:
0010111001010001
1011101000110100
1011111001110101

The schema or similarity template for these individuals is:
#01#1#100##10#0#
Where # represents "don't care" (in effect a mismatch).

The schema theory then looks at the following: " The length of the binary string, L, (in this case 16); " The defining length which is the distance between the first gene which is not represented by a # and the last gene which is not represented by a # (in this case 15-2=13) and " The order of the schema, which is the number of characters containing a 1 or 0 (in this case 9).

Within a given member of a population, each gene can be represented by one of three characters (#,0 or 1) and therefore the number of schemata present within the member is 3^L. If the population contains N members, then the total number of schemata within the population is $N3^L$.

The schema theory states that having a large number of schemata within a population increases the probability that high quality building blocks (i.e. areas within a chromosome that represent good solutions to the problem being considered) will be formed. Therefore, the theory says that long, binary chromosomes will tend to give better solutions. This is linked to Holland's (Holland, 1975) thoughts on implicit parallelism within genetic algorithms.

The theory further states that binary strings have a greater information carrying capacity than a real number representation and Holland presents arguments to support this but they contain assumptions about the similarity of representations that would be used when comparing real number and binary encoding that are not necessarily valid.

There are also difficulties that occur when using binary encoding. The most obvious is that there is usually a need to translate the binary representation (the genotype) into a form where it can readily be understood by human beings (the phenotype). Also using binary can lead to excessively long chromosomes which can be difficult to handle within a computer. Finally there is the so called Hamming cliff problem.

Taking a simple example where the answer to a problem is 7

[0111] = 7 BUT [1000] = 8 i.e. Completely different in binary format

This means that for a genetic algorithm to converge on a solution from 7 to 8, the schema would be completely different (note the problem doesn't occur between 8 and 9). This is what is called a Hamming Cliff - in other words all the zeros have to change to ones and vice versa. The problems with this can be partially overcome by using Gray Scale coding (Table 1) template for table placement

Table 1. Binary and Gray Scale Encoding

Decimal	Binary	Gray
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0010
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111

The arguments in favour of real number representation are less complex than those given above, see Parmee (2001). One of the main advantages is that they completely avoid Hamming cliffs and therefore cope much better with dealing with convergence type problems. However, the main argument in favour of real number representation is that it has been proven to work well on a wide variety of problems.

2.3 Fitness Assessment - The Fitness Function

As stated above, the processes within a GA can be thought of as being similar to those in Darwin's theory of evolution where individuals which are better suited to a given environment, on average, stand a better chance of survival and hence of passing their genes on to the next generation. It is therefore necessary within a genetic algorithm to have some sort of function that represents the environment. That is, a function that measures how good a chromosome is in terms of it being a possible solution to the problem being considered. There is an immediate challenge that arises here and that is that for many problems, very little is known about the form of the search space. This can cause difficulties for traditional optimization

methods but genetic algorithms are able to find good solutions without needing detailed information regarding the problem being solved. Thus the performance assessment function for a genetic algorithm is not referred to as an objective function, as would be the case for traditional optimization methods, but instead, as a fitness function.

The fitness function is always problem specific so it is hard to give a general description and instead an example will be given.

Take for example the problem shown in Figure 3 (Griffiths and Miles, 2003). A load is to be applied uniformly to the top of a space and there is to be a support at the base of the space. In between, some form of support is needed. If the third dimension is included, this could be a problem to find the optimum beam cross sectional area say for a simply supported beam.

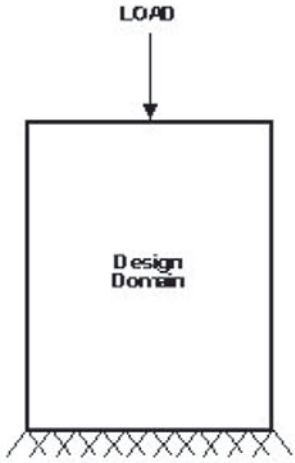


Figure 3. The Design Domain

The encoding of the problem is as shown in Figure 4 where the space is split up into squares (called voxels) and each voxel is represented as a gene in a binary string (see the upper part of Figure 4). If material is present in a voxel, it is represented as a 1 and if the voxel is void it is represented as a 0. Thus in Figure 4, the expected answer for this loading case is an I beam as shown.

However, the challenge for a genetic algorithm would be to generate an I beam from an initial population in which the members are created by a random number generator. One of the phenomena to which a beam is subjected is bending, so the fitness function should contain a function which deals with this. Therefore a cross-sectional shape is required that keeps normal stress levels, those caused by the application of a bending moment, within the allowable stress limits while utilising

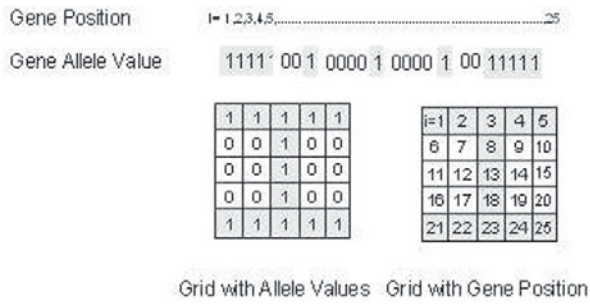


Figure 4. The Problem Encoding

the minimal amount of material. Minimising the required material arrives from a desire to reduce self-weight. For this initial solution only bending moments are considered with the load being applied uniformly and symmetrically to the upper surface of the beam and the support being provided in the same manner at its lower surface.

Such a load case produces normal stresses over the entire cross-section of the beam which vary in intensity depending upon the shape of the cross section and the location of material within it. Each solution is evaluated utilising two standard stress analysis equations. The stress value for an individual active voxel can be calculated using the bending stress (Gere and Timoshenko, 1997):

$$\sigma = \frac{My(i)}{I} \tag{1}$$

where M is the applied bending moment (Nmm), y(i) is the distance of the ith active voxel from the neutral axis in millimetres, and I (mm^4) is the second moment of area with respect to the horizontal neutral axis.

The second moment of area for the shape (with respect to the neutral axis) is calculated as Gere and Timoshenko (1997) shown below:

$$I = ^n \Sigma_{i=1} (y(i)^2 A) \tag{2}$$

where y(i) is distance of the ith voxel from the neutral axis of the shape in millimetres, A (mm^2)is the area of a voxel and n is the number of active voxels.

The neutral axis is assumed to be a horizontal line that passes through the centroid of mass of the shape. The voxel representation system applied in this research reduces the design space to a series of squares of uniform size and density. Therefore it is acceptable to calculate the neutral axis as being the average position of active voxels (average position of material).

$$NeutralAxis = \frac{\Sigma Y_{base}}{Active} \tag{3}$$

where Ybase is the distance of the material from the bottom of the design space in millimetres and Active is the total number of active voxels.

It is evident from a simple analysis of these equations that the distance of an active voxel from the neutral axis is most significant in determining the optimal shape for the cross-section. Increasing the average value of y, will increase the second moment of the shape and decrease the maximum value of voxel stress. Normal stresses require material to migrate towards the vertical extremities of the design space where the normal stress is greatest (forming two symmetrical flanges). A valid solution to this simplified mathematical model would be any shape that does not exceed the stress constraint, with the best solutions being those that satisfy this constraint with the minimum amount of material. As only normal stresses are currently being considered, there is no requirement for the genetic algorithm to develop a web (primary shear stress carrier) or any other means of connecting the two elements.

The fitness function has been designed to minimise the area of the beam (active voxels) within the maximum allowed stress. Equation (4) describes the hard constraint (i.e. must be satisfied) for this problem as follows:

$$SVoxelMax \leq \sigma_{max} \tag{4}$$

The actual fitness function is given in the following equation. As can be seen, it is designed to encourage the development of minimum mass, high stress solutions which meet the imposed constraints. The scaling factors are present within the function to allow for the different magnitude of the various terms and also to ensure that each effect is given an appropriate level of importance within the function.

$$Fitness = \frac{1000}{(Active + (\frac{1}{SVoxelMax} + A(P1) + B(P2) + C(P3))} \tag{5}$$

where Active is the number of active voxels and SVoxelMax is the maximum stress value of any voxel.

$$P1 = (SVoxelMax - \sigma_{max}) \tag{6}$$

$$P2 = (SVoxelMax - SVoxelMin) \tag{7}$$

$$P3 = (NumberofExposedVoxelSides) \tag{8}$$

$$A,B,C = ScalingFactors(A = 100, B = 30, C = 10) \tag{9}$$

σ_{max} is the maximum stress value permitted in N/mm^{2}within the material and SVoxelMin is the minimum stress in any of the voxels.

The variable Active is used in a way which promotes solutions which have the minimum amount of material as is P3. The other terms in the fitness function encourage the efficient use of material.

The fitness of an individual is inversely proportional to the number of active voxels (required material), and hence the search favours minimum self-weight. To promote efficient use of the material, and minimise redundant stress carrying capacity, (1/SvoxelMax) is applied so that solutions operate at the most efficient stress values for the material utilised for the cross-section. P1 is applied to solutions that violate the maximum stress constraint. P1 is the only hard constraint at this point, and therefore violations are penalised heavily. P2 again encourages the genetic algorithm to evolve solutions that operate at high stress levels to promote efficiency. P3 is applied to minimise the development of small holes and isolated voxels by penalising solutions relative to the number of exposed voxel sides. P3 also aids the development of solutions with minimal material as it requires voxels to form tightly packed shapes for optimality. Each isolated voxel or small hole will result in a P3 value of four (four exposed sides), thus encouraging the GA to group together sections of material.

Scaling factors A, B and C are used to increase the weight of each variable, a standard practice when applying evolutionary searches. Scaling has been applied to prioritise the satisfaction of constraints over criteria. During initial generations the scaling factors for constraint violations are set relatively low to allow the genetic algorithm to explore the greatest range of potential solutions. During the final generations, the scaling factors are increased to ensure the final solution meets all constraints. Additionally, the scaling factors are applied to ensure that no single variable dominates the search. This is necessary as the range of values for each variable varies significantly. For example, during final generations, Active has an average value of 750 per solution, where as P1 tends towards +/- 3. These penalties can ensure all constraints are satisfied simultaneously, without any undue bias. Scaling factors are also applied to the criteria elements of the fitness function to ensure no one criterion dominates another.

So what sort of results does this fitness function give? The answer depends on factors other than just the fitness function but Figure 5 shows typical solutions evolved by the genetic algorithm after the 2000 generation.

Table 2 highlights the details of the best evolved solutions, at the end of a 2000 generation run (60000 evaluations, i.e. a population size of 30 chromosomes). The tests were conducted with a design space of 300mm by 175mm and an applied bending moment of 200,000,000 Nm and a maximum allowable stress of 100 MPa. In addition to near optimal shapes being achieved (within 5 voxels of the known optimum), the solutions are within the stress constraint limits. The genetic algorithm has a success rate of approximately 90 per cent at locating near optimal solutions.

Figure 5. Results from 3 separate runs achieved with two dimensional genetic operators, for bending only so no webs

Table 2. Results of tests conducted by Griffiths and Miles (2003)

Run -	Percentage Active Voxels	Bending Stress N/mm^2	Surface Area of Exposed Voxel Sides	Optimal Shape Achieved
Initial Population	-	Average 155	Average 1925	Not Applicable
1	34.38	100.01	172	Yes
2	34.52	99.09	179	Yes
3	34.42	99.85	175	Yes

The above example shows how for a relatively simple problem, the fitness function can be fairly complex and also how it typically contains a mixture of objectives and constraints. However, as can be seen from Figure 2, there are still many other aspects of a genetic algorithm to be considered and without careful consideration of these, the above results would not have been possible.

2.4 Selection

Having determined the fitness of each chromosome (i.e. potential solution to the problem), the next stage is to select from the population those chromosomes that will go forward to the mating pool. The mating pool is then used to breed the chromosomes of the next generation so those individuals that are not selected will not be able to pass on their genetic information to the next generation. There are various methods of selection used with genetic algorithms but the basis of nearly all of them is that the chances of the fitter individuals being selected for the mating pool is stronger than that of the less fit.

There are two common methods that are used, the roulette wheel and tournament selection.

With the roulette wheel, a conceptual wheel is constructed, with each chromosome being given a slice of the wheel which is proportionate in size to its fitness.

So a fit individual is given a relatively large slice and a less fit individual a smaller slice. The wheel is then "spun" (in reality a random number generator is used) with the result being an angle between 0 and 360. The selection operator determines which chromosome's slice covers the resulting angle and that individual is then passed through to the mating pool. The wheel is spun sufficient times to provide enough individuals for mating. Note that it is entirely possible for an individual to be selected more than once.

The tournament selection method is simpler but generally more effective. In its simplest form two individuals are selected from the population and the fitter of the two is allowed to go forward to the mating pool. More complex forms select a greater number of individuals but the process is still the same with the fittest going forward.

With both these selection methods, it is entirely possible that the fittest individual may not be selected for the mating pool. To overcome this problem, many implementations of genetic algorithms use some form of what is known as elitism where the fittest individual is automatically placed either in the mating pool or it is passed directly to the next generation. Sometimes rather than just the fittest individual, the fittest X percent (say 10 percent) are chosen. Some people argue that elitism is a bad thing and in such cases the usual procedure is to filter off the fittest individual from each generation and keep them on one side in a separate location so that if the individual is the fittest produced by the genetic algorithm during the entire run, the information is not lost.

2.5 Crossover

Having chosen the chromosomes that are to be bred to form the next generation, there are two main operators that typically form the breeding process. The first of these is crossover and this is intended to mimic the processes that occur in nature when two individuals come together to create a new individual. In genetic algorithms, there are many forms of crossover, the simplest of which is single point.

The process starts with the selection of two individuals from the mating pool. For convenience, we will assume that these are both binary strings as follows:

[1000100100001] [0111100011001]

Using a random number generator, a point is chosen some where along the length of the two chromosomes (say between the 5th and 6th bits) and the strings are then cut at this point and the new chromosomes formed by taking the left hand end of the first individual and combining it with the right hand end of the second and vice versa. The result is as follows:

[1000100011001] [0111100100001]

Thus two new individuals are formed from the two parents. The process con-

tains a significant degree of randomness about it. There is deliberately no attempt to determine whether or not this is a good point at which to cut each individual because although in this generation the result may not be advantageous, it may result in another generation or two in some further recombination which will give good results. This is both a strength and a weakness of genetic algorithms. Their ability to generate new and interesting results enables them to search widely throughout the problem space and as the search starts to converge on certain areas of the search space, the crossover occurs between high fitness individuals and so should result in similarly fit individuals. However, towards the end of the process, when convergence is almost complete, crossover can be very disruptive and at this stage it can in some circumstances be advantageous to stop the genetic algorithm and instead apply a more traditional, deterministic algorithm such as hill climbing.

There are many other forms of crossover. The basis of most of these is the use of a mask as shown in Figure 6. This shows how a mask can be used for one point,

Figure 6. Crossover Mechanisms

two point and multi-point crossover. In effect the mask is another binary string of the same length as the chromosome. In each position on the mask is either a 1 or a 0. If there is a 1, then that part of the chromosome remains unchanged. If there is a 0, then the corresponding bit from the other parent is imported into the chromosome. This gives the basis of how crossover works but as will be show later in the section on topological reasoning, the choice of what mechanism to use can have an impact on the ability of the algorithm to search effectively.

2.6 Mutation

As mentioned above, there are two commonly used operators with genetic algorithms. Crossover is used to combine attributes from the two parents but in nature there is also another mechanism at work and this is mutation. This in nature allows species to change their characteristics over time and hence introduce features that were not present in the original population. In genetic algorithms, the working of mutation is very similar. Within the original population, there is only a limited amount of genetic information and while crossover can combine and mix this information in new ways to try and find desirable results, it is possible that from time to time, the introduction of new material will be useful. This is the basic thinking behind the mutation operator.

Mutation has to be used carefully but if this is done, then its impact can be highly beneficial. If there is no mutation, then often the result is that the genetic algorithm fails to find the best areas within the search space. However, if too much mutation is allowed, then its impact can become disruptive and the process can degenerate into a random search.

The typical method of applying mutation for a binary chromosome is to define a so called mutation rate (typically about 0.01) and then generate a random number between zero and one for each bit. If for a given bit, the number is less than the mutation rate, then the bit is "flipped"; that is a 1 is changed to a 0 or vice versa. To show how this works an example is given below. For simplicity, a population of just 3 chromosomes will be used, with each chromosome containing 4 bits. The original population is:
[1010] [1110] [0010]
For each every bit of each chromosome a random number in the range zero to one is generated as follows:
0.801,0.102,0.266,0.373
0.120,0.096,0.005,0.840
0.760,0.473,0.894,0.001
A mutation rate is then applied. This is typically a value that is fixed at the start of the execution of the genetic algorithm and it has to be in the range of zero to one (the same as the random numbers). Any bit whose random number is less than or equal to the mutation rate is then bit flipped (i.e. a one is altered to zero and vice versa). The mutation rate is typically set at a low value, otherwise the search degenerates. For this example, the mutation rate is set at 0.008. As can be seen only two numbers are less than 0.008 so only two bit flips occur to give:
[1010] [1100] [0011]
With real number chromosomes, mutation is slightly more complex but works on similar principles. Typically two types of mutation operator are used with real numbers, jump and creep. The former allows a given number in a chromosome to change to any value within the allowed range and the latter just makes a small

change. The way that this is then applied is to initially generate a random number between one and zero for each member of the chromosome and if the number is below the mutation rate, the member is selected for mutation. A further random number between one and zero is then generated for each member. If the number if below say 0.5, then the jump operator is used and if it is equal to or above 0.5, then the creep operator is used.

Referring to the work of Bradshaw (1996) on power stations, for each member of a chromosome, the possible range was between zero and 100 percent. In practice the upper limit was set at 80 percent to allow for maintenance time, so the jump operator allowed the number to be mutated to any value between 0 and 80. The value to be applied was again determined by using a random number generator. The creep operator allowed the number to be change by 5 percent, again with the value being determined randomly. For this particular problem, it was found to be advantageous to apply the mutation operators equally at the start of the search process (i.e. below 0.5 use the jump operator and equal to or above 0.5 use the creep). However as the search progressed, the jump operator was found to be too disruptive when applied at a high rate and so the rates were progressively changed as the search progressed, so that in the later stages, the jump operator was rarely applied.

2.7 Inversion

There is another operator that is sometimes used. In this the entire order of the chromosome is swapped so that the right hand end becomes the left hand end etc. So for example the chromosome [00001111] would become [1111000]. In practice, inversion is rarely used because it is generally found to be too disruptive.

2.8 Convergence

With most problems that are tackled using genetic algorithms, information about the final answer is completely lacking and therefore it is not possible to say, with absolute certainty, when the search process is complete. In such circumstances, it is usual to let the search progress for a pre-determined number of generations. However, some means of checking that the answer obtained is probably somewhere near the best possible is necessary. One way of checking for this is to plot a graph of the change in the fitness that occurs as the search progresses. Typically the values plotted are for the fittest individual within a population, although it is also usual to plot the average fitness of the population as well. An example plot of the fittest individual is given in Figure 7. In this particular case, the correct answer to the problem is known and it is given a fitness of 100.

As can be seen from the Figure 7, the process gets close to the best answer sometime around the 300th generation and then slowly converges so that by the

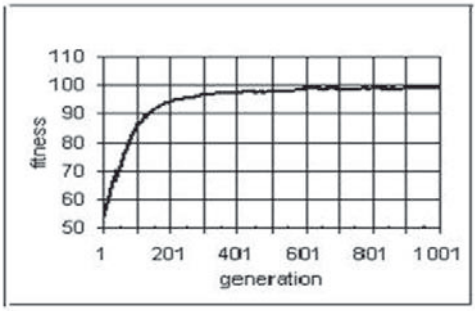


Figure 7. An example of change in fitness by generation

600th generation, it finds the best answer. This is for a case where the answer is known, but for examples where the answer is not known, one would hope that the change in fitness will follow a similar path although to some extent this is a function of the problem space.

In cases where the problem space is reasonably continuous, then one would expect the behaviour to be similar to that shown in Figure 7. However, even when the process follows a reasonably smooth curve, it is not possible to guarantee that the answer obtained is a good one and sometimes a sudden jump occurs in the fitness curve. This for example could be when mutation takes the search to a different area of higher performance.

Where the problem space contains significant discontinuities, then one would expect a plot of fitness against generation to be much less smooth than that shown in Figure 7. In such cases, it is much more difficult to judge with confidence when the process seems to have converged.

There is an alternative approach to the above which has been developed by Parmee (1998). His approach has been specially developed for design problems where rather than finding "the best" answer, one is more concerned with learning about the problem space and finding areas of high performance. The details of Parmee's method are too complex to be explained here but the overall concept is that at each generation, chromosomes that pass a pre-determined level of fitness are copied to a separate area where they are stored and at the end of the search process, the contents of the area are presented to the user using a variety of techniques including box plots and hyper planes. The plots can be presented in a variety of formats including variable space and objective space plus the fitness landscape. The approach is sophisticated and powerful and is recommended to anyone who wishes to progress beyond the basic techniques that are being dis-

cussed here. There is also a simpler and less powerful technique that is sometime employed using Pareto fronts. Good examples of the application of this to structural engineering are to be found in the work of Khajehpour and Grierson (1999) and Grierson and Khajehpour (2002).

2.9 Summary

The above covers the basic techniques for developing a so called canonical genetic algorithm. There are many variations on the techniques described but for getting started and learning about genetic algorithms, it is recommended that initial work uses the above approaches and then develops further complexity as necessary. The additional techniques are to be found in the literature. Also, there are other very useful algorithms in addition to genetic algorithms and again once the basics of evolutionary computation have been mastered, it can often be beneficial to try other procedures.

3 Using Genetic Algorithms for Design

3.1 Introduction

This section will build on the description of a canonical genetic algorithm and show how the techniques can be used to solve problems in design. The problem that will be examined is the conceptual design of typical commercial buildings such as multi-storey office blocks.

3.2 Building Design

In structural terms, a typical commercial building such as an office, a hospital or a hotel, consists of columns and beams with the floors being either concrete or composite slabs. The columns and beams are usually either steel or concrete, although sometimes timber can also be used. The design of such a building is undertaken by a team of designers which includes architects, structural engineering and building services engineers. These people are brought together at the behest of who ever is commissioning the building (typically called the client) to interpret the client's needs and produce a satisfactory design.

Design can be thought of as consisting of a number of stages. The very earliest stages go by various names but typically are called client briefing or brief development. Briefing starts with the client expressing what sort of building they want, what they will use it for and what their overall parameters are in terms of cost and the areas required. There then follows a series of conversations between the client and one of more of the designers to develop these basic ideas into a more comprehensive statement of what is required. At this stage, typically there are no drawings other than a plan of the area in which the building is to be located (i.e. a

site plan).

The next stage of the process is then called conceptual design. This is where most of the basic decisions regarding the form of the building are taken and typically by the end of this process, 80 per cent of the costs of the building are fixed so it is vital that the right decisions are made at this stage. Generally during conceptual design, the details of the design have yet to be fixed and so often decision making is based on rough estimations and heuristics rather than absolute values.

Following conceptual design, the overall form of the building is fixed and there then follow a series of stages in which detailed calculations are undertaken to check out the performance of various components. These typically involve numerical analysis techniques such as finite elements and finally the detailed drawings and other contract documents are produced.

In this section, the discussion will focus on the conceptual design of a building and how and why this should be supported by appropriate computational techniques. However before this discussion starts, one very important ground rule has to be established. In all complex decision making processes, it is vital that the decisions are made by human beings, not computers. Computers are excellent at processing and handling large amounts of information but they do not possess knowledge about the world (other than that which has been specifically encoded within software) and they lack common sense and judgement. So what has to be aimed for in any decision making system, is a process where the human being and the computer working together are able to produce a result which is superior to that which can be produced by other techniques. To achieve this the process must make the best use of the strengths of both the human being and the computer.

Typically when undertaking the conceptual design of a building, each design will look at several options. For example, the structural designer will look at a steel and concrete option and possibly they may look at one or two variations of each of these. So typically each design may look at as many as 6 possible options. Khajehpour and Grierson (2003) estimate that for the design of a typical building, at the conceptual design stage, there are at least 400,000 possible choices from which the design team can choose, so if each designer looks at 6 options, there are a lot of possibilities that have been ignored. Of these 400,000, it is possible that some will be poor choices that any experienced designer would automatically reject. Also some may be very similar but nevertheless, the evidence is overwhelming that unless they are extremely lucky, the design team will manage to do no more than produce a design solution that satisfies the constraints, a so called satisficing solution.

So what is needed, is something to enable the design team to find good solutions from the range of possible options and it is here that genetic algorithms can be of use. The following description is based on the work of Sisk (1999). The system is called BGRID. The discussion will first look at the features of BGRID

before then taking the reader through a typical user session.

3.3 Representation

The representation can be thought of as the choice of features of the problem that are to be included in the chromosome and the encoding method (real number or binary). It is usually advantageous to keep the representation as simple as possible. The BGRID chromosome contains 4 types of information:

- The (x,y) coordinates of each column centre. As the number of columns varies between solutions this means that BGRID does not have a fixed length of chromosome.

- The structure-services integration strategy. This describes how the services (i.e. ventilation, plumbing, electricity, etc) are incorporated with the structural elements above the suspended ceiling. There are 3 options within BGRID, these being, separate, partially integrated and fully integrated. The option chosen affects the grid because for each option there are limited number of structural solutions, which in turn dictates the economical column spacings. The choice of structural system influences the floor to floor height and hence has a major impact on cost.

- The environmental strategy. There are 3 options within the system, natural ventilation, mechanical ventilation and air conditioning. The choice of these is influenced by factors such as the building location and building depth and they have a major impact on the height of a building and hence its cost.

- The final component of the genotype is the clear floor to ceiling height. The overall height of a building has significant cost implications and also dictates whether or not it can be illuminated using natural daylight.

Real number coding is used. Each column is represented by two integers, one for the X and one for the Y co-ordinate. The structural services and environmental strategies are represented by integers and the floor to ceiling height is a real number. A rather short example of a genome is given in Table 3. The left hand part, consisting of 6 numbers, contains the X column coordinates, the middle part, consisting of the next 6 numbers, contains the Y column coordinates and the right had end contains the other parts of the genotype.

Table 3. An example genome

0	25	50	75	90	100	0	20	40	65	80	90	0	1	2.9

3.4 Reproduction, Crossover, Mutation

In BGRID only the fittest member from a given generation is automatically passed through to the next generation. Roulette wheel selection is used to determine which chromosomes are to be passed through to the mating pool.

The crossover mechanism has to take account of the structure of the genome which is split into 3 distinct segments. To avoid creating a lot of illegal solutions in crossover, the crossover operator is applied separately to each of the 3 parts of the string. Single point crossover is used for each part, with the actual location within the string being chosen randomly. If crossover produces column spacings which do not fit the constraints imposed by the various grids, the nearest suitable value is determined and the illegal value altered to conform. Where crossover produces a chromosome where the values are out of order (i.e. not all the values increase from left to right) the ordering is adjusted to suit. Also a further check is run to determine the column spacings. Spacings which exceed 18m (the maximum spacing allowed by the structural systems used in BGRID) are reduced by inserting another column. Where columns are too close, one column is removed.

The mutation operator uses a mutation rate of 0.01. For the genes which contain the column coordinates the range of mutation is restricted, as is explained in the following example:

For the gene selected for mutation, the first process is to look at the values of the 2 genes on either side of it (obviously for genes at the end of a segment one can only look in one direction). So taking for example the gene shown in table 1. If the 3rd gene with a value of 50 is assumed to be the one that is to be mutated, it is necessary to ensure that the mutated value is consistent with the various grids that have been defined for the building. So the mutation process starts by setting up an array of possible values which are consistent with the grid spacings and fit in between the 2 adjacent column spacings (in this case 25 and 75m). The new value is then chosen randomly from the array of values. If for this example it is assumed the modular grid spacing is 1.5m and the smallest preferred grid dimension is 3m, then an array of grid spacings can be generated that starts at 28m and goes in 3m increments to 72m. A value is then chosen at random from the array (say 55m) and this becomes the new value. For the 3rd part of the genome, the values are mutated in the normal way.

3.5 The Fitness Function

The development of BGRID was undertaken in close collaboration with practising designers. A fitness function that allows the user to search the design space in different ways to examine the impact of various constraints is something, which the designers identified as being a vital feature. Such features which allow the user to operate the system in a flexible manner, allowing a full exploration of the design

space while leaving the final decision to the designer, are a desirable feature of any design software. For the BGRID fitness function, this flexibility is achieved by allowing the user to alter the weights of the individual criteria within the function. This feature also allows for the different types of users (e.g. architects, structural engineers, etc.) to alter the bias of the search to suit their own discipline specific viewpoint. In more detail, the fitness function works as described below.

The first part of the fitness function checks the hard constraints. If these are not satisfied or acceptable, the implications are serious and the associated solutions will be penalised. The checks conducted for these constraints include:

- Check if overall height is less than the height restriction (if present);
- Check if the design option is compatible with the chosen structural system, particularly with respect to span lengths.
- Check the uniformity of grid (this has a significant influence on buildability)

If the above are not satisfied a penalty function is applied to the individual's fitness. The value of penalty varies for each constraint. The fitness of the individual design solution is multiplied by the inverse of the penalty function with the penalties being as follows:

Height

$$PF1 = 1; IF height > height restriction \tag{10}$$

$$PF1 = 0 : IF height \leq height restriction \tag{11}$$

Design Option Suitability

$$PF2 = PF2 + 0.25 \tag{12}$$

Uniformity

$$PF3 = PF3 + 0.25 \tag{13}$$

The way the above are applied is as follows:

- For the first penalty function, its value is 1 if the constraint is violated and the overall height is greater than the height restriction and 0 if the overall height is within the set limits.
- For the second penalty function, the value is initially 0 and then is increased by increments of 0.25 for each bay that doesn't fall within the economic span range of the structural design option.
- The third penalty function also increases by increments of 0.25 for each different bay dimension, (e.g. if there are three bay sizes of say 9, 12 and 15m within the grid, a penalty function of 0.5 will be applied. If all the bay sizes are the same, then the penalty function is zero).

The three penalty functions are then added thus:

$$PF = PF1 + PF2 + PF3 + 1 \tag{14}$$

The second part of the fitness function deals with the soft constraints. Soft constraints are more difficult to quantify and the way this is dealt with within BGRID is that each individual is assessed relative to all the other individuals generated up to that point. That is, the worst and the best examples of each criterion are used to determine how good the design solution is for this particular aspect of the design. This is achieved as follows:

$$FIOBJ = \frac{FI - FIBAD}{FIGOOD - FIBAD} \tag{15}$$

where: FIBAD=the value of the worst individual generated up to that point, FIGOOD=the value of the best individual generated up to that point, FI=the value of the evaluated parameter for the individual under consideration.

The objective is to maximise the value of FIOBJ. Each of the three components below can be weighted by the user to reflect their importance for a particular search strategy. This is the main mechanism by which the user influences the search. The weight factors range from 0 to 4, with 0 being unimportant and 4 being highly significant. The three components are:

- Large clear span,
- Minimising cost,
- Minimising environmental damage.

The Large Clear Span component enables the user to search for solutions with as large a span as is feasible, within the constraints of the design. Often clients prefer buildings with as much column-free space as possible because of the greater flexibility of such configurations.

When the user opts for the Minimum Cost being significant, the system doesn't actually optimise on real costs but features which are cost significant, these being:

- Total weight per floor area (kg/m^2)(Including steel, slab, deck, reinforcement, services)
- Overall building height (m)
- Net/gross floor area ratio

The aim is to minimise the first two factors and maximise the third. It was felt by the designers who collaborated in the development of BGRID that such an approach, rather than using quantities and costs, was sufficiently accurate at the conceptual design stage to guide the search towards low cost solutions.

A similar approach is used for the Minimising Environmental Damage component with the following aspects being considered:

- Depth of space,
- Clear floor-to-ceiling height
- Location

The depth of space is determined by the dimensions of the floor plate (making allowances for atria etc.) A higher than normal floor to ceiling depth is required if natural daylight and natural ventilation are to be viable options but if a building is situated in an urban environment the only ventilation strategy that is viable is air-conditioning.

The above formulae are summed to give an overall fitness as follows:

$$FI = \frac{\Sigma(FIOBJ.WF)}{\Sigma PF + \Sigma wf} \tag{16}$$

where: FIOBJ = sub-fitness of individual for each evaluating parameter (from equation XX), WF = User determined weighting factor for the given evaluating parameter, PF= penalty function for the hard constraints.

This determines the raw fitness of each individual in the population and is used to determine the rank order. The Standard Fitness Method, see (Bradshaw and Miles, 1997) is then used to assign pre-determined fitness values which effectively determines the degree of selective pressure assigned to each individual. The pre-determined fitnesses are calculated using an algorithm based on the normal distribution and are then used to determine fixed slot sizes on a roulette wheel. The individual with the largest rank order (i.e. raw fitness) is then given the largest slot, the next in rank order the next largest etc.

3.6 BGRID: A Typical User Session

As with any design system, BGRID requires some initial information from the user regarding the building which is to be constructed. The data input is divided into four sections:

- Geometrical information (plan dimensions and number of floors). The design is restricted to rectangular floor plans. The user is asked for X and Y dimensions and the number of stories. During the search process, BGRID will almost certainly produce some solutions which do not exactly conform to the required plan dimensions but which are high fitness solutions. These are not penalised by the fitness function as it is assumed the designer would wish to be aware of them.
- Site location. The user is given 3 options to choose from, urban, suburban and rural. Also the user is asked to specify any overall height restrictions on the building (usually imposed as a planning requirement) and the maximum

desirable floor to ceiling height. The above factors have a significant bearing on the available options for lighting and heating and ventilation.

- Location of architectural spaces, i.e. cores and atria (Figure 8). The user is first presented with a screen which asks for the maximum occupancy (in terms of number of people) of each storey and from this a minimum width of fire escape can be calculated using the rules from BS 5588 Part 3. The process then moves onto fixing the number and location of cores and atria (if any). In the search process, BGRID may, in a given individual, make slight adjustments to the sizes of cores and more often to atria, to better fit them to the structural grid.
- Dimensional constraints (see below)

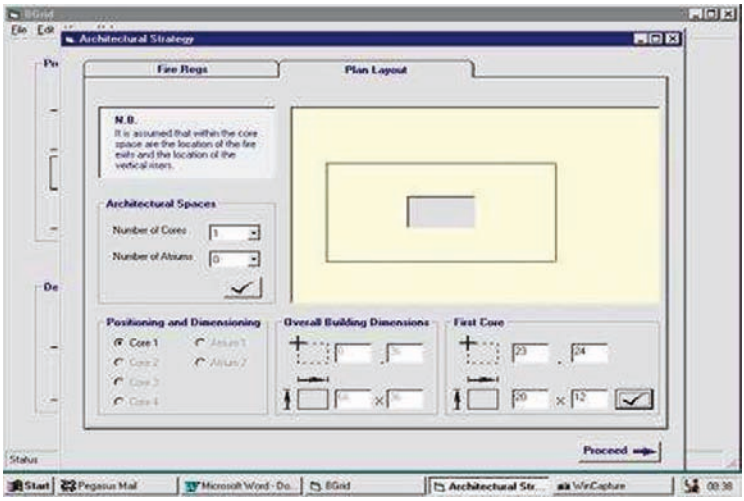


Figure 8. Specification of Core and Atria Spaces

The dimensional constraints determine the grids that can be generated for the genetic algorithm's initial population. The user is asked to input the various grid dimensions, which are to be used by the system. The user chooses from a menu which contains all the available grid sizes and these are then used within BGRID as follows.

The building dimensions are built up in multiples of the initial modular (i.e. constructional) grid. Planning must take into account apparently insignificant elements, such as fluorescent tubes, as their cumulative impact can have a significant effect on the layout of the building. Based on advice from the designers who col-

laborated in the BGRID development, the user is offered a choice of initial modular grid dimensions of 1200, 1500 and 1800 mm.

The next dimension is the critical grid, this being the absolute minimum distance that can separate two adjacent columns. This measurement is determined by the intended use of the building. Again, based on advice from the industrial collaborators the available critical grid dimensions within the system are 2400, 3000 and 3600mm.

Following this, the user inputs the preferred minimum bay dimension (i.e. the preferred distance between two adjacent columns). The users are also given the opportunity to specify the maximum preferred distance between two adjacent columns. These constraints are then used to limit the form of the grids generated for the initial population, to ensure that they meet the designers' requirements. This results in a massive reduction in the search space, which is both an advantage and a disadvantage. The benefit is that the problem is much less complicated to solve but occasionally the search space is so constrained that the search is unable to find solutions which are significantly better than the best of the initial (i.e. random) population. The available minimum and maximum bay dimensions within BGRID are given in Table 4. Again these have been provided by the industrial collaborators.

Table 4. Building Bay Dimensions

Available Minimum Bay Dimensions (mm)	Available Maximum Bay Dimensions Bay Dimensions (mm)
4500	6000
4800	7200
5000	8000
5400	8400
6000	9000
7200	10000
8400	18000

The next part of the system allows access to all the background information, such as the various structural configurations and section sizes used by the system to generate the design solutions. Various default sizes are used within the system but the user is free to change all these as is described below.

3.7 Creating the Initial Population

In the initial population for each individual the bay dimensions are chosen randomly although obviously account has to be taken of the overall dimensions input by the user and the ranges of available values are restricted based on the

dimensions of the grids which have been indicated by the user. The main challenge is to generate column layouts which do not violate the constraints. The other parameters are likewise generated randomly using values from the available predetermined ranges.

3.8 Component Sizing

BGRID sizes structural components using span/ depth ratios. This level of detail was felt by the collaborators adequate for conceptual design, although subsequent work in this are (see below) has used a far more accurate approach. The span/depth ratios have been obtained from manufacturers' catalogues. As with the section sizes, the user is free to override any of the system's decisions regarding section sizes and span depth ratios and they can, if wished, insert their own. The provision of facilities that enable the user to access and adjust the assumptions and defaults within BGRID is a deliberate strategy. The collaborators asked for a system which was both transparent and flexible. A typical example of the information provided for a given structural flooring system is shown below in Figure 9.

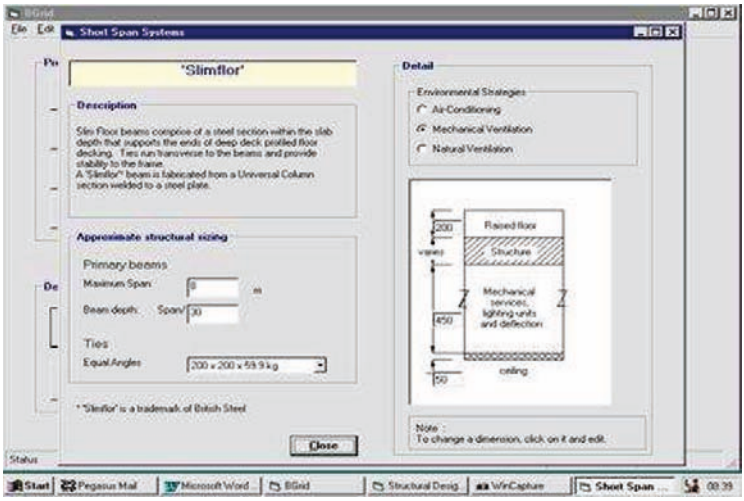


Figure 9. Short Span Structural System Background Information

Currently BGRID contains information regarding 3 types of flooring system. For the short spans there is the Slimflor system (Slimflor is a trademark of CORUS plc.). In this system, the structural and services zones are fully separated. The Slimflor beam is integrated within the steel deep deck, thus minimising the depth of the structural zone. This is advantageous for a highly serviced building, al-

lowing complete freedom for the horizontal distribution of services. Each of the environmental strategies requires different depths of zones for items such as air conditioning and ventilation and BGRID contains suitable defaults for each strategy. The user is free to change any of these default depths if this is necessary. The ties, which provide frame stability, can be selected from a list of angles provided within BGRID. These dimensions are then used to calculate the overall height of the building. With all the flooring systems, BGRID contains values of minimum and maximum allowable spans.

For the medium span system, a composite steel beam and concrete slab system is provided with the structural and environmental zones being partially integrated. The default vertical dimensions for each ventilation strategy are also shown within the system and as with the Slimflor, all default information can be changed if necessary.

For the long span system, a steel stub girder and composite slab system is included. The maximum span for this system is 18-20 metres. Three typical grids exist and these are fully documented within BGRID. The grid descriptions include details of the bottom chord, secondary beam and overall depth of the structure. Again, the user is free to change all of these values. For the long span system, the services are fully integrated within the structural zone. Further information on the flooring systems is given in Sisk (1999).

3.9 Controlling the Search

The next part of the user interaction concerns the search process and how to guide this by altering the fitness function weights. As described above, the fitness function contains 3 components, the aim of these being:

- Minimising cost
- Minimising clear span
- Maximising use of natural resources

BGRID allows the user to weight the importance of each component using weight factors, which range from 0 (irrelevant) to 4 (extremely important). The next step is to activate the genetic algorithm, which generates an initial population of 50 solutions randomly within a confined search space. As explained above, each solution contains information about the column grid, the structural system, the environmental strategy and details of the vertical dimensions of the building. This information makes up the genotype of the genetic algorithm. The genetic algorithm runs for 50 generations with the whole process taking approximately 20 seconds. Fifty generations has been found to be more than sufficient for all cases tested to date.

3.10 Search Results

The user is able to access information regarding maximum, average and minimum fitnesses for each generation. Details of the 'best' solution for each generation are provided in both textual (Figure 10) and graphical form, the latter showing a floor plan with column spacings and positions of cores and atria (if present).

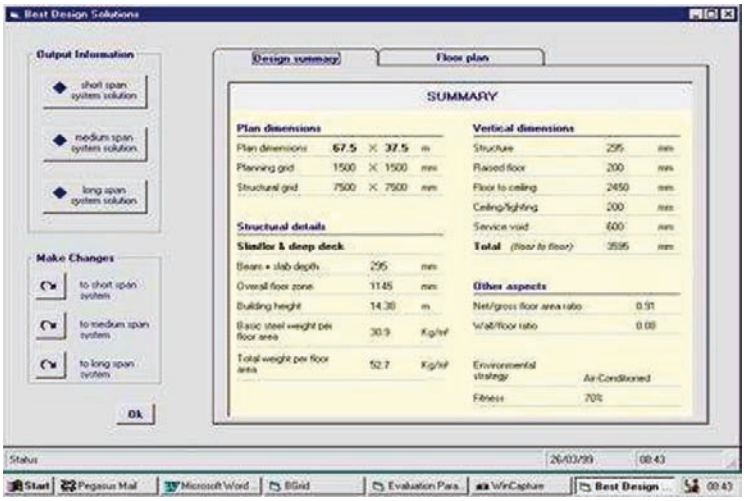


Figure 10. Text Based Design Summary

Also, the system provides the best design solution in both graphical and numerical form for each of the short, medium and long span structural systems. The planning grid, the structural grid, the vertical dimensions, overall height of building, weight of steel and total weight of floors are provided, as is the net/gross floor area, the wall/ floor area ratio, the environmental strategy and the fitness. The user is able to edit the design solution, for example moving the position of the cores or adding an atrium. BGRID automatically checks any amendments to ensure that none of them violate the constraints and re-evaluates the design working out a new fitness value. This allows the user to see how beneficial their changes are.

3.11 Evaluation

BGRID has been developed in collaboration with practising designers and has been evaluated throughout its development, by architects, structural engineers and services engineers. Although in the earlier stages some teething problems were identified, in general, and especially towards the end of the development process, the response to BGRID has been very positive. Evaluators appreciated the flexible

way in which BGRID could be used to investigate design problems. Also they believed that the level of detail, both in terms of the design and the underlying calculations, was correct for conceptual design although as stated above, further work by the author has used more accurate approaches. One of the advantages of using computationally assisted conceptual design, is that the level of accuracy can be increased, thus reducing risk.

3.12 Further Developments

One feature of BGRID that is very limiting is that it can only cope with buildings with rectangular floor plans. This deficiency has been looked at by Shaw et al. (2005b) who developed a genetic algorithm based method for designing buildings with orthogonal boundaries. To achieve this polygon partitioning techniques are used to decompose a floor plan into rectangular sections. This further development of BGRID is called OBGRID In terms of data requirements, the main difference between BGRID and OBGRID is that for the latter, the user has to fully define the shape of the building. As the shape is more complex than just a single rectangle, this is a somewhat more involved procedure for OBGRID. However as OBGRID breaks all areas down into rectangles, the discussion will start by looking at how OBGRID deals with a rectangle. The basic chromosome for a rectangle is identical to that used for BGRID (see Figure 11 and Table 3) with the first part containing the X column spacings, the second the Y column spacings and the third part the structural system, services integration / environmental strategies and the floor to ceiling height.

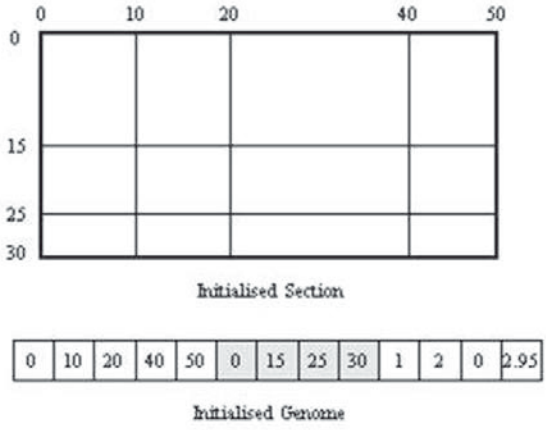


Figure 11. Example initialised, rectangular genome

3.13 Initialising a Rectangular Genome

Rectangular floor plans are initialised by considering the genome's three sections (Figure 11):

- Section 1: starting at the upper left hand corner of the floor plan (it is always assumed that the top left hand corner has the local coordinates (0,0)) the algorithm generates random column spacings in the x direction until the end of the floor plan is reached.
- Section 2: restarting at the upper left hand corner, the algorithm generates random column spacings in the y direction until the end of the floor plan is reached.
- Section 3: The final section is initialised with randomly selected variables.

Unlike BGRID, no effort is made to constrain column positions to realistic spacings. This is to make the GA search for solutions in both the feasible and infeasible regions and hence improve the search. However, the fitness function does penalise individuals that contain a range of column spacings. This is to encourage a degree of uniformity in column spacings, which aids 'buildability'.

3.14 Evolutionary Operators

Genetic algorithms search the solution space by using biologically inspired operators. However because the genome is divided into 3 distinct sections and variable length genomes are used, the evolutionary operators have been amended to reflect this:

- Mutation: used to inject new solutions into the population improving the search by (hopefully) prevent premature convergence (Goldberg, 1989). Having selected an individual's genome a new value is generated for a random gene. If the mutation operator selects a gene from sections 1 or 2 (Figure 11), the gene is replaced with a randomly generated value between 0 and the limits of the floor plan. Unlike BGRID, which restricts the new spacing to a value between the two adjacent genes, OBGRID simply generates a random spacing and, if required, sorts the genome so that the column spacings increase from left to right. If a gene from section 3 (Figure 11) is chosen, it is mutated as normal.
- Recombination: used to exploit the information already in the population. OBGRID employs a single point crossover operator. Single point crossover is used because it is simple to implement even with variable length genomes (as in OBGRID). However rather than applying the crossover operator on the whole genome, it performs a separate crossover on each of the genome's three sections and happens with BGRID.

3.15 Selection

OBGRID uses the tournament selection technique (Goldberg, 1989), which has been found to give better performance than the roulette wheel method used in BGRID.

3.16 Fitness Function

The fitness function assigns a single numerical value to an individual that reflects how 'good' it is (although a multi-objective fitness function might be more appropriate, however the goal of this work is to develop a representation). The basic form of the fitness function is the same as that used in BGRID although OBGrid uses a quadratic penalty function, which assigns a greater penalty to a larger transgression.

3.17 Illustrative Example: Rectangular Building

The following test case was designed to assess OBGrid's performance.

- Building dimensions: 60m x 18.2m,
- Height restriction: none,
- Population Size = 100,
- Maximum Number of Generations=50,
- Tournament Selection (size=2),
- Elitism used,
- Probability of reproduction = 0.1,
- Probability of Mutation = 0.3,
- One point crossover,
- Crossover probability = 0.6,
- Real encoding
- Random initialisation with no seeding.

The fitness is based on the overall height (Lower is better but with constraints), column spacing compatability and column spacing uniformity.

3.18 Results

The performance graph (Figure 12) shows the fitness of the best of generation during the run, while (Figure 13) indicates the optimum layout. Note that unlike

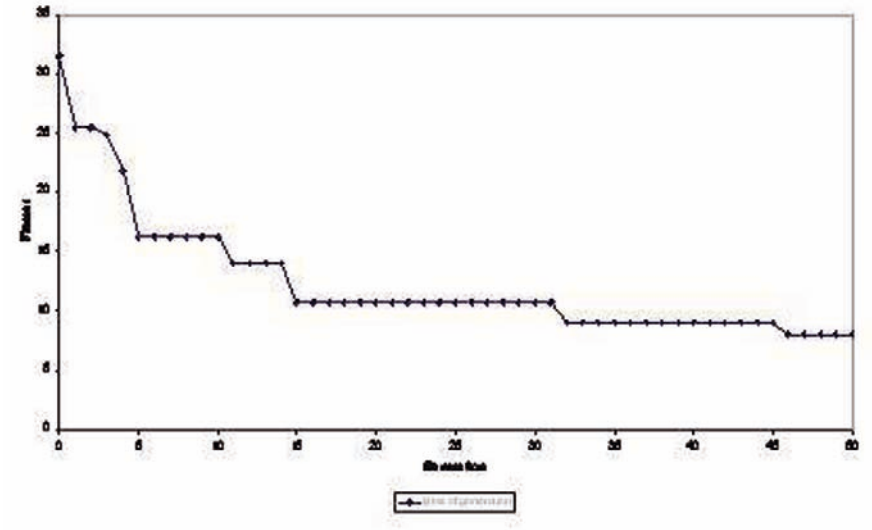


Figure 12. Fitness progression for OBGRID

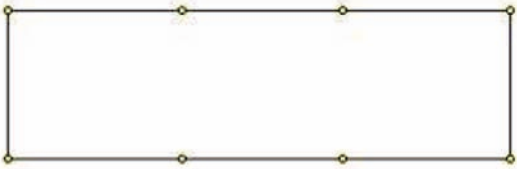


Figure 13. Rectangular Building Results for OBGRID

the example given above of a plot of convergence, in this case the improvement in fitness occurs in distinct steps rather than being a smooth curve. This reflects the nature of the search space. The best solution was found in generation 46 and proposes using the long structural system with spacings of 20m (n.b. the maximum length restriction applied on BGRID is not imposed here) and mechanical ventilation.

3.19 OBGRID and Orthogonal Buildings

Having shown how rectangular buildings are dealt with, the discussion will now look at how OBGRID partitions an orthogonal floor plan into rectangles and

then uses the rectangular methodology described above to design a layout. However an additional complication is the need to ensure column line continuation throughout the building, which is achieved by using an 'adjacency graph'.

3.20 Polygon Partitioning

Computational geometry (Shamos, 1978) is the study of efficient algorithms (usually computer based) and data structures for solving geometrical problems. The partitioning of polygons is a major topic in this field and several algorithms have been developed. However a 'sweep line' approach was considered the most appropriate for column layout design. Sweep lines techniques (Rourke, 1998) move an imaginary line, the 'sweep line', over a polygon from top to bottom or left to right. During a sweep, the line is stopped at 'event points' when the polygon is partitioned. In this work, event points are any reflex vertex on the boundary (Figure 14). Partitioning is completed in two stages:

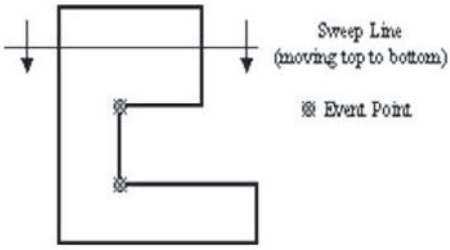


Figure 14. An example sweep line

- First stage: a line is swept from top to bottom. When the line encounters an event point, it extends the relevant boundary edge horizontally across the floor plan until it encounters another edge and splits it, at the point of intersection. This partitions the building into several, 'thin' rectangles (the left hand side of Figure 15).
- Second stage: a line is swept from left to right across the boundary, further partitioning the rectangles created by the first stage: creating a grid pattern (the right hand side of Figure 15). It should be noted that for each floor plan, there is a unique partitioning. Therefore once an orthogonal floor plan has been partitioned, no further partitioning is required (during the search).

An adjacency graph is used to ensure column line continuity throughout the building and to tell the genetic algorithm which sections of the building are connected to one another.

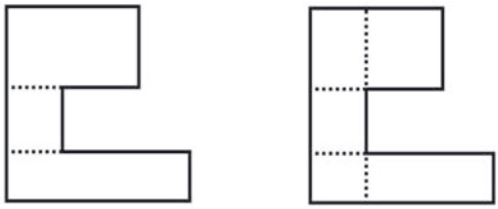


Figure 15. The Two Stages of Partitioning

With the floor plan decomposed into a grid of rectangles, each partition must now share at least one edge with another partition (with an upper limit of four). These edges become vital during the evolutionary process because they will be used to prevent the genetic algorithm from generating nonsensical solutions. To monitor the status of neighbouring sections, the partitioned polygon has an 'adjacency graph' associated with it (Figure 16). Associating a node with each partitioned section and linking it to an adjacent section create the adjacency graph.

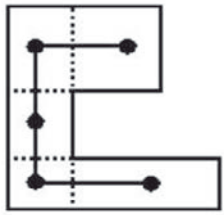


Figure 16. Adjacency Graph

Once a floor plan has been decomposed into a series of rectangular partitions and an adjacency graph constructed, each partition has a genome associated with it. Conceptually, each individual contains a set of genomes: with each genome representing a rectangular partition linked by the adjacency graph. With the building partitioned and adjacent partitions monitoring each other, a genome can be generated for each partition. The initialisation process starts by selecting the left most upper partition (this is an arbitrary selection as the initialisation process could theoretically start at any partition, however to improve the algorithm's 'transparency' it always starts at the same location). As the overall dimensions of this partition are known (and that it is a rectangle) the algorithm uses the initialisation procedure described above. Having initialised the first partition, the algorithm selects an adjacent partition and generates a new genome for it. However as the next partition

must share a common edge, the algorithm firstly copies the column spacings for this edge. For example, in Figure 17, left hand side, the x and y spacings from section 1 are copied into the adjacent partitions. New spacings are then generated for the remaining directions (Figure 17 right hand side). In complicated buildings it is possible that both directions have been initialised, in this instance the partition simply copies spacings from adjacent sections and does not generate any new spacings. By constantly maintaining and updating the status of neighbouring sections,

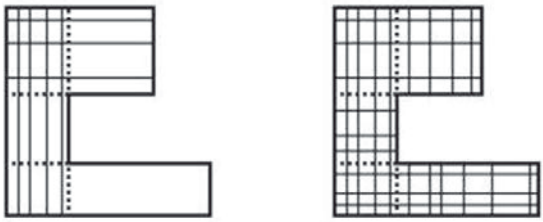


Figure 17. First and Final Stages of Partitioning

via the adjacency graph, the algorithm ensures continuity throughout the building. This continuity is vital to prevent the building from becoming a series of blocks that when placed together do not form a coherent solution. For example, in Figure 18 when considered in isolation each section is valid however, when considered as a whole, the building's layout is flawed because the columns do not align. The

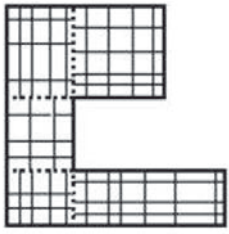


Figure 18. Invalid Partitioning

third section of the genome is assumed to be fixed throughout the building therefore every genome has an identical section 3.

3.21 Evolutionary operators

The same evolutionary operators described previously are applied to each rectangular partition. However, to ensure column continuity some additional steps are applied:

- Mutation: Having selected the individual to mutate, the mutation operator randomly chooses, with uniform probability one partition of the building. It then selects an individual gene and generates a new variable for it. If the mutation operator selects a gene from sections 1 or 2, the gene is replaced with a randomly generated value between 0 and the limits of the section (Figure 19). Unlike BGRID, that restricts the new spacing to a value between the two adjacent genes, the new system simply generates a random spacing and, if required, sorts the genome so that the column spacings increase from left to right. Having altered its genome, the section is placed back into the building and all adjacent sections are updated (Figure 19). This final step means the mutation operator is able to modify the building in only one location but the change ripples throughout the building preventing column alignments degenerating.

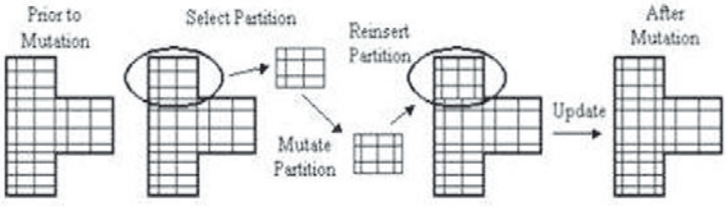


Figure 19. Mutation

- Recombination(i.e. crossover): OBGRID employs a single point crossover operator (Goldberg, 1989), which exchanges part of the genomes associated with a section of the building. However once recombination has been accomplished, the altered sections are reinserted into the building and all other adjacent partitions updated (as with the mutation operator described above) (Figure 20).

3.22 Fitness Function

OBGrid applies the same fitness function as BGrid to each partition in the floor plan and aggregates the results.

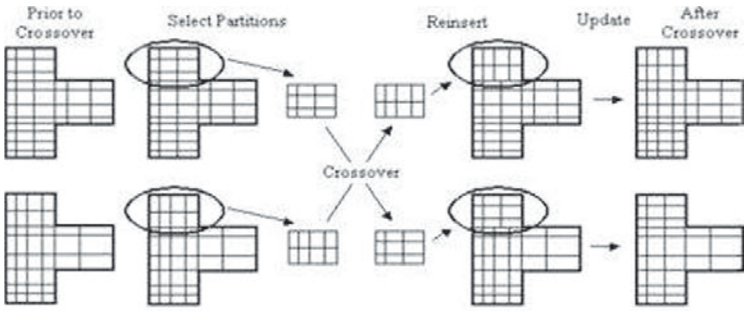


Figure 20. Crossover

3.23 Illustrative Example: An Orthogonal Building

The following test case was designed to assess OBGrid's performance. The basic floor plan is given in Figure 21 and the partitioned floor plan is shown in Figure 22. The parameters used are as for the rectangular building example.

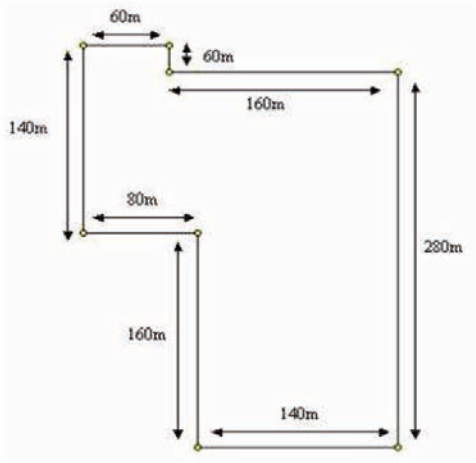


Figure 21. Orthogonal Example: Floor Plan and Dimensions

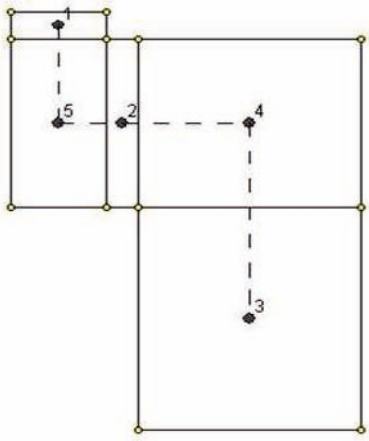


Figure 22. Partitioning for Orthogonal Example

3.24 Results

The performance graph (Figure 23) shows the fitness of the best of generation during the run, while Figure 24 indicates the optimum layout. The best solution

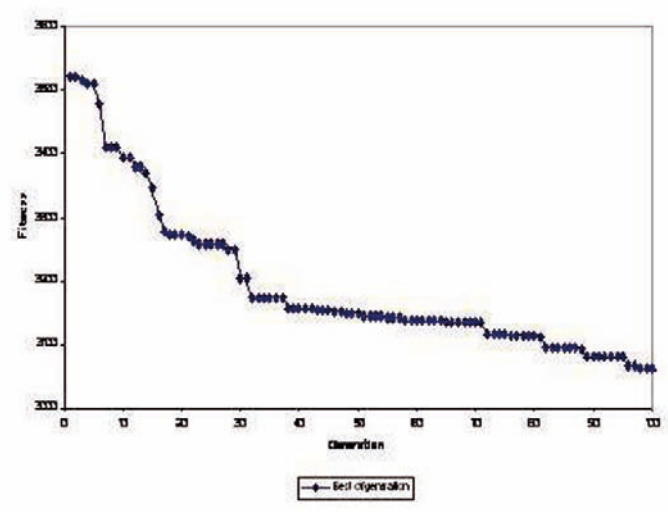


Figure 23. Performance Graph

was found in generation 97 and proposes using the long structural system with an average column spacing of 20m and mechanical ventilation.

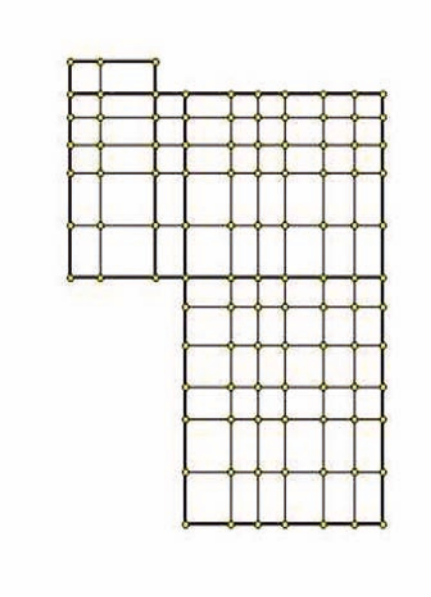


Figure 24. Best Solution: Generation 97

4 Genetic Algorithms for Design

In the above examples, although the form of the building was a factor, it was not the main purpose of the search which was more aimed at finding a structural layout, once the floor plan had been fixed. However in many examples of Engineering design, finding the best form for a given set of criteria, is an important factor. This process goes by the general name of topological reasoning and in the following section the discussion will look at some of the more salient methods that are used in conjunction with genetic algorithms. In particular, the discussion will focus on the methods of representation (i.e. what should be in the chromosome) used with topological reasoning for structural design.

4.1 Parameter Based Representation

In the above examples for building design, the chromosome contained the parameters which defined the column locations. Where the topology of what is to

be design has already been fixed, the use of parameters is appropriate and useful but for topological search, using parameters is very limiting and generally should be avoided. The reason for this can be shown by the simple example given in Figure 25. In a domain where the topology of the solution is unknown, then a

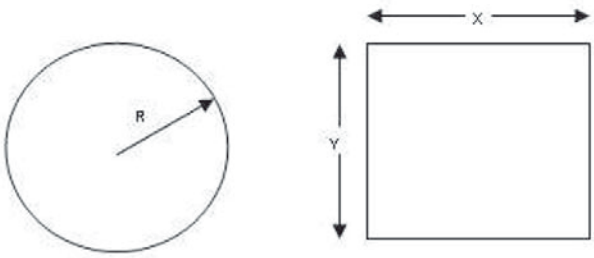


Figure 25. Parameter Based Representation

representation which allows the algorithm to describe a rectangle would, using parameters, require an X and Y dimension. However to also allow for a solution where the answer is a circle would require a radius. One can of course go on adding further possible shapes and for each one there would have to be a separate parameter set and the algorithm would have to be able to recognise each one. It becomes even worse if one considers a population of possible solutions with different topologies and then considers how one would cope with typical operators such as crossover. So for search problems where the topology is a part of the search, the use of parameters is very restrictive and almost certainly will lead to a poor solution. So other forms of representation are required.

4.2 Ground Structures

Another form of possible representation that has been used extensively in the past is ground structures. With a ground structure, in theory, the complete range of possible components and topologies is pre-defined and the idea of the search is to determine which of the components can be removed to just leave ”the best” structure. In practice, it is impossible to predict in advance all the possible components and topologies and therefore, right from the very start, one can say that ground structures will not allow a full search of all the possible solutions. Ground structures are mostly used when a truss is the preferred form. A ground structure then consists of a truss with nodes in fixed locations. Each node is initially connected to all the other nodes by a structural member. The ”optimisation” process consists of selectively removing structural members and searching for that com-

bination which best satisfies the search criteria (Figure 26). The method has the attraction of being simple but by fixing the number of nodes and their location, the search is severely constrained and so the resulting structure is unlikely to be the optimum for the given criteria.

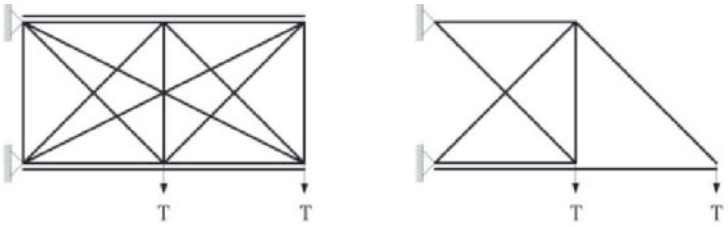


Figure 26. An example of so called optimization using a ground structure. On the left is the ground structure and on the right is the ”optimised” structure (Deb and Gulati, 2001)

4.3 Graphs

Various forms of graph based representation have been used with genetic algorithms. As with ground structures, their application has mostly been to trusses although they can be used for other layout problems such as building floor plans. Graphs allow one to model the connectivity between nodes while varying the location of the nodes (Borkowski and Grabska, 1995). This gives them a distinct advantage over the ground structure method and generally for problems which involve linear and connected elements, a graph representation is very good. Yang (2000) presents a simple method of using a tree based structure to describe trusses (Figure 27 and Figure 28). In mathematical terms, this structure can be expressed as:

I=((e1,m,i),(e2,m,k),e3,i,l),(e4,k,n2),e5,i,n1),e6,k,p),e7,i,n4),
(e8,p,j),(e9(l,n3),e10,p,n3),e11,j,n2),e12,j,n1))

Where the ”e” values are the ”edges” (i.e. the lines connecting the nodes in the graph). In the above case, the edges are given zero values but they could easily contain properties such as the cross sectional area of the structural component, etc. Such a structure as that given above can easily be turned into a genetic algorithm chromosome. When applying operators such as crossover, there can be problems with the resulting structures being unviable (for example mechanisms) and so sometimes ”repair operators” are needed. Graphs are useful way of representing structures such as trusses which consist of many interconnecting members but have only limited applicability to other forms of structures.

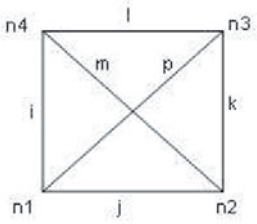


Figure 27. An Example Truss Structure

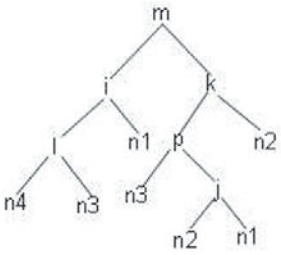


Figure 28. Graph Based Representation of Truss Structure

4.4 Voxels

The use of voxels has already been discussed above and an example is given in Figure 29. For problems where there is a relatively high amount of solid material in comparison to the amount of voids, then voxels can be a useful technique. However, some of the problems are shown in Figure 29. This is an extension of the pure bending problem discussed previously, where in this case some allowance is made for shear forces, resulting in the formation of a web. The initial population consists of randomly created solutions and from these one hopes to generate suitable structural forms. The shape on the left shows a solution from an early stage of the process. It can be seen that there are a number of isolated sections of material. As the load is applied at the top and supported at the base, there is no way that these can form any part of the main structure. However, at a later stage when being combined with another chromosome using crossover, these isolated voxels may link up with other voxels and form part of a useful structure.

Figure 29. Potential Problems and Eventual Success using Voxels

So there are two competing forces. The first is to get rid of the isolated voxels because they serve no immediate purpose and the second is that they may form a useful building block at a later stage. The solution is that there has to be a compromise and the fitness function has to contain some functions which assist with the removal of isolated voxels (such as measuring the total length of the exposed voxel edges) which however are not so punitive as to immediately remove them all.

Another problem is for intermediate solutions, such as that shown on the left and in the centre, if these are structural forms, how does one analyse the performance of such cross sections so that one can ascertain their fitness. Griffiths and Miles (2003), used approximate methods. More recent approaches by the author have used finite elements but this leads to long execution times.

Crossover and Mutation with Voxels

With a voxel representation, there is no difference between the genome and the phenome. So, unlike most genetic algorithm problems where crossover and mutation operate on a parameter set which have no direct connection to the phenome, with voxels the operators directly change the form of the solution. This makes the choice of operator a matter of vital importance. As discussed in the preceding chapter, the usual crossover mechanism uses a mask and the components are swapped directly.

For the problem shown in Figure 29 above, the problem is essentially two dimensional and so mask based crossover in two dimensional form works as shown below in Figure 30. However, experience has shown that using this form of

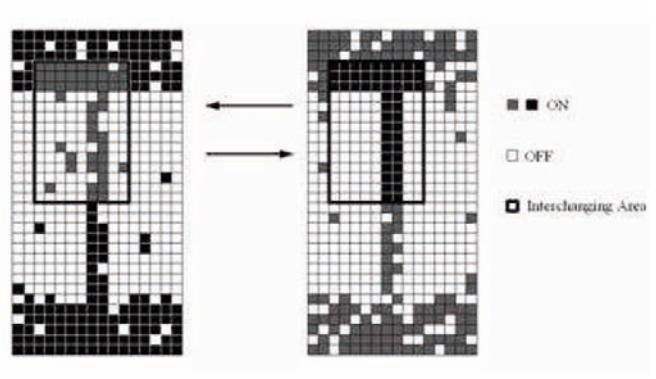


Figure 30. Two dimensional crossover using a conventional mask (Zhang and Miles, 2004)

.

crossover with voxels does not give a satisfactory solution. The problem can be seen in Figure 30 where once the solutions start to converge, the material which is swapped is almost identical and so the ability of the algorithm to search for new solutions is severely constrained. Extensive experience based on the work of Griffiths and Miles (2003) and Zhang and Miles (2004) has shown that what is needed is a two-dimensional crossover with swap as shown below in Figure 31 where the areas to be exchanged are chosen randomly. This may at first sight seem more disruptive than the normal crossover mechanism and therefore intuitively less likely to give satisfactory answers but extensive testing has proved the superiority of the swap operator.

Similarly with mutation, just changing one or two bits in a large search space (e.g. 32 x 64 voxels = 2048 bits) has little impact. Griffiths and Miles found that it was better to mutate on an area basis, say selecting a group of 4 voxels which form a square and applying mutation to these.

4.5 Designing Geodesic Domes Using a Computational Geometry Based Representation

Geodesic domes are structural space frames with regularly spaced members which are typically arranged in a triangular format. The search for good solutions to the design of such domes is an interesting problem because of their 3 dimensional nature. Pure geodesic domes have homogeneity in both member length and nodal angular incidence and also have a geometry that is usually based upon the sub division of a spherical surface into triangles, these being the simplest non-deformable rigid shape. Geodesic domes are considered by some to be

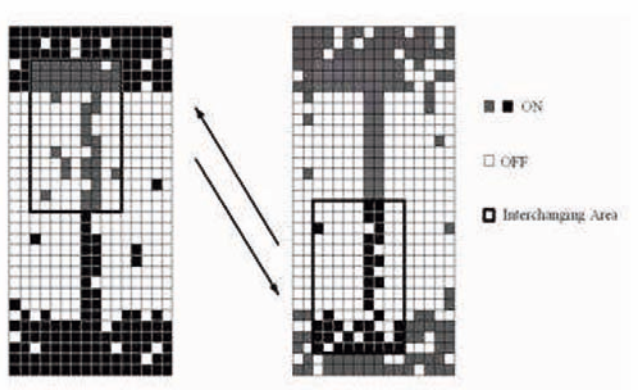


Figure 31. Two dimensional crossover with swap (Zhang and Miles, 2004)

the strongest, lightest and most efficient building system Motro (1994). In this section, supporting the design of geodesic like domes by means of a genetic algorithm is discussed. The domes that are designed not pure geodesics because the constraints on member lengths and angles are not imposed although these could be implemented by introducing extra constraints. The aim is just to produce viable and efficient domical structures.

The representation used is based on a technique known as convex hulls which are a subsection of computational geometry. Convex hulls are useful for this sort of work because the dome can be represented as a collection of vertices which are then joined by lines to form a series of triangles. From the point of view of evolutionary computation, the method has a further attraction that the vertices can be generated randomly and the algorithms associated with convex hull will then determine which vertices lie on the exterior and thus form the convex hull itself. Additionally, this ability to find the exterior vertices and ignore the others means that should a vertex become displaced from the convex hull, possibly due to the impact of operators such as crossover, it does not have to be deleted but can remain within the population and may be of use in a succeeding generation.

In the following, the discussion concentrates on the implementation of the representation. For further details about this work see Shaw et al. (2005a). The initial sections contain the description of the computational geometry techniques which have been used with the later sections describing how these are implemented in a GA.

Convex Hulls

Computational Geometry is the study of algorithms and data structures for solving geometric problems (Shamos, 1978). For the dome design problem, a

sub-section of computational geometry called convex hulls is used. The convex hull of a finite set of points S, is considered to be the convex polyhedron with the smallest volume that encloses S. Convex hulls can be used to form polyhedra with a polyhedron being defined as a three dimensional object composed of a finite number of flat faces, edges and vertices (Figure 32). It can also be described as the 3D generalisation of a 2D polygon. Within this work, every polyhedron will be convex with triangular faces and referred to as convex polyhedra. Polyhedral

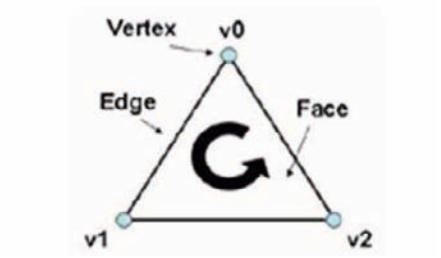


Figure 32. CCW

Figure 33. Tetrahedron with CCW

faces have one important feature, they maintain their vertices so that when seen from the exterior they appear in counter clockwise (CCW) (Figure 33). This ensures that the right hand rule (Figure 34) always yields a vector normal to the face, pointing away from the polyhedron (Rourke, 1998).

Signed Volumes

To calculate the volume of a tetrahedral from the vertices, one has to use the determinant form of the cross product (Rourke, 1998). The volume is described as 'signed' because it can be positive or negative with a negative volume being defined as being generated when a face f forms a tetrahedron with a point p that

Figure 34. Right Hand Rule

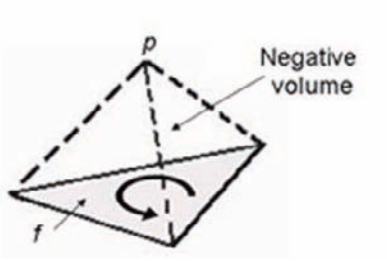


Figure 35. Negative volume generated by CCW face f and point p

can see its vertices in a CCW manner (Figure 35). A face f is considered to be visible from some point p, iff a line drawn from p to some point x interior to f does not intersect with the polyhedron, denoted as CH, at any point other than x. Visibility can be also be formally defined by using sets (17). It should be noted that (17) defines a face that is 'edge on' to p to be invisible.

$$iffpx \cap CH = \{x\} \tag{17}$$

The visibility of a face f from a point p can be determined by calculating the signed volume of the tetrahedron defined by f and p. f is considered to be visible from p, *iff* the signed volume is negative.

Incremental Algorithm

There are several algorithms which can be used to construct a convex hull (Rourke, 1998) (de Berg, 2000). In this work, the incremental algorithm is used. This constructs the convex hull CH, of a finite set of points S, by initially developing the convex hull CH_{i-1} of a simple sub-set S_i of S. Having constructed the convex hull for S_i, the algorithm then adds one point at a time from S to S_i updating

the convex hull as it progresses.

For example, consider a finite set of points

$$S = \{p_{(}1), p_{(}2), p_{(}3).....p_{(}n)\} \tag{18}$$

The convex hull is initialised using a tetrahedron which is defined by four points taken from S. The base of the tetrahedron is formed by 3 non-collinear points and a fourth point at its apex, non-coplanar with the first three (if S does not contain these points, then it is a 2D set and invalid for this problem).

The next remaining point pi in S, is then added to the existing convex hull CH_{i-1} by considering the question: Are there any existing faces of CH_{i-1} visible from p_i?

- No. If none of the existing faces of CH_{i-1} are visible from p_i, then pi must be internal to CH_{i-1}. Therefore CH_{i-1} is still valid, as it encloses all points in S_i and remains unaltered. This is one of the strengths of this form of representation because it effectively ignores vertices which are in an inappropriate position. This avoids the need for repair algorithms and other features which are commonly implemented with topological search problems to deal with the disruption that can be caused by crossover.
- Yes. If some of the existing faces of CH_{i-1} are visible from p_i, then p_i must be exterior to CH_{i-1}. Therefore CH_{i-1} is invalid, because it no longer encloses all points in S_i and must be updated to include p_i.

Updating the Convex Hull

The convex hull CH_{i-1} is updated in two stages: locating the horizon and incorporating the external point. An external point p_i divides the existing hull CH_{i-1} into two regions: the visible and the invisible. These two regions are separated by a curve called the 'horizon' (de Berg, 2000) which is formed by the series of edges that are adjacent to a visible and invisible face (Figure 36 a). Therefore once the visibility status of every face from p_i has been determined, the horizon can be located. The external point is incorporated into the hull by appending a set of new faces to it that have p_i as a vertex. The new faces are triangular and constructed from a horizon edge with an apex at p_i (Figure 36 b). After building these new faces, the original faces that were visible from p_i are now underneath (the new faces) and should be deleted along with any superfluous edges and vertices. At the end of this process a new convex hull is complete (Figure 36 c).

Considering the definition of visibility that states that 'edge on' faces are invisible, then any new faces will be appended to these existing 'edge on' faces. However, if 'edge on' faces are considered to be visible then the algorithm will attempt to remove them and replace them with a single new face. Unfortunately

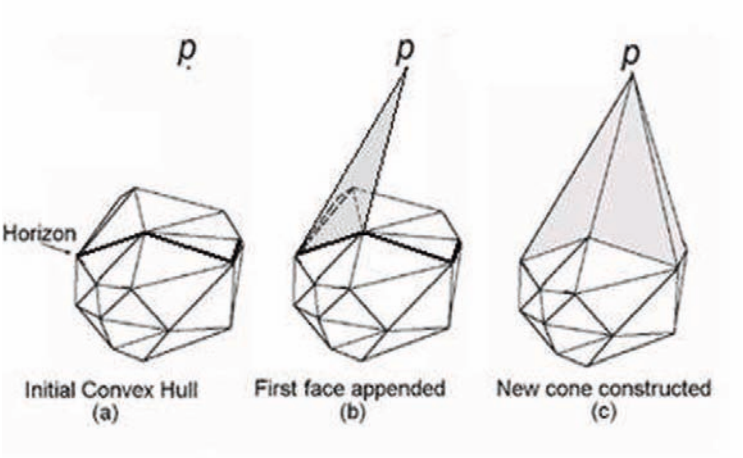


Figure 36. Updating an existing hull (adapted from Rourke (1998)).

if the new point to be inserted (into the convex hull) is coplanar with the existing 'edge on' faces, the new face may not be triangular or result in the existing face fracturing into a series of smaller faces and thus make the algorithm significantly more computationally intensive (de Berg, 2000).

4.6 An Example

This work uses a GA to search for potential solutions. Real number coding is used. The genome used in this work is subdivided into three sections (Figure 36):

- Location of and magnitude of loads: The loads represent forces that must be supported by the structure in addition to its self weight and are included in the genome because it is assumed that a load can only act at a node (nodes are solely constructed from the information within the genome). Therefore although the user defined loads are constant for all individuals, they are included in the genome to add transparency (the self weight is calculated once the dome has been constructed and is not included in the genome).
- Location of the dome supports: Dome supports represent locations at which the dome is attached to the ground or supporting structure, for most domes these are vertices in the plane z = 0. These can user specified or searched for during the evolutionary process.
- Location of dome vertices: Dome vertices represent nodes in the dome. For non-trivial structures, this is the largest section of the genome. These are initially generated by the incremental algorithm.

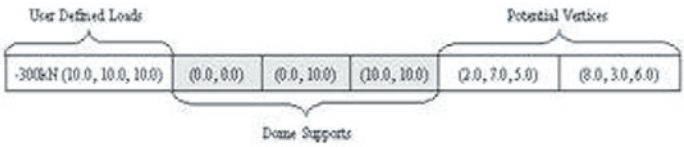


Figure 37. Example genome

Representation

Each section of the genome is represented in a different way :

- User defined loads: As the user defined loads are constant, the evolutionary operators do not need to modify them.
- Dome supports: A pair genes (i.e. X and Y coordinates) represents every dome support: (It is assumed they are all in the plane z=0).
- Potential vertices: A triplet represents each node location (i.e. a separate gene for the x,y and z coordinates).

When the incremental algorithm constructs the convex hull, the final structure is dependent on the order in which the vertices are added. So two convex hulls constructed from the same set of points but with different orderings, could produce structures that have identical vertices but different arrangements of faces and edges and thus different structural responses. Therefore, genome ordering can be significant.

Initialisation

At the start of the search process, the user only has to input the location of the loads and define the size of the circular base. The user can also, if required, specify the number and location of the dome supports. If this isn't done, the algorithm will search for appropriate support positions during the run. Where not specified, support locations are initially a series of randomly generated points on the circumference of the circular base generated by selecting two numbers x_1 and x_2 from a uniform distribution between -1 and 1 (ensuring that the sum of the square of both numbers is not greater than or equal to 1). The corresponding Cartesian coordinates related to x_1 and x_2 are given by equations (19) and (20) (Weisstein, accessed 2005).

$$x = \frac{x_1^2 - x_2^2}{x_1^2 + x_2^2} \tag{19}$$

$$y = \frac{2x_1^2 x_2^2}{x_1^2 + x_2^2} \tag{20}$$

Vertices are generated from random points within a cube that is centered on the dome's base and with a side length of three times the diameter of the base. This procedure is used to prevent the GA searching in completely unproductive regions, while at the same time not biasing or inhibiting the search. To prevent additional supports being generated, vertices may not lie on the domain boundaries. At the outset each individual has a random number of vertices in its genome. However because the dome is only constructed from vertices that lie on the convex hull, it does not necessarily follow that all of these will be used to construct the dome.

4.7 Evolutionary Operators

Within the GA's search process, the loads section of the genome is unaffected (as these loads must be carried by every solution) while the crossover and mutation operations are applied to the remaining sections individually. Selection is achieved using a standard tournament approach. An 'n point' crossover operator is employed which is allowed to alter the length of the genome. However, crossover must ensure that the second section always contains gene pairs and the third section contains gene triplets. Two mutation operators are used in this system: point and shuffle. Point mutation randomly selects a gene to alter and then uses the same procedures as described during initialisation to generate a new point depending on whether a support or vertex is selected. Shuffle mutation reorders a length of the genome. This operator is included because genome ordering is important thus a solution maybe improved by shuffling the genes.

4.8 Fitness Function

This work uses the minimisation of structural weight, enclosed volume and surface area as its major objectives. These are combined with a structural parameter that seeks to ensure constraints such as allowable buckling, tensile and compressive stresses are not violated in the structure. The structural analysis module uses the *Trussworks* package (Bucciarelli and Sthapit, accessed 2003) that allows users and analyse 3D structures using the Direct Stiffness Method. This is computationally much faster than the more rigorous analysis methods such as finite element analysis and more than adequate for what is basically a conceptual design tool. To search for the optimum number and location of supports, the GA initially generates a random number of supports and uses overall weight and stress constraints to guide it. This is because for every additional support there must be at least two additional structural members. These increase the overall weight: while the removal of a support increases the loads carried by each of the remaining structural members which may violate a structural constraint. Both of these scenarios reduce the individual's fitness and hence the algorithm is guided towards good solutions.

When calculating an individual's fitness, the 3D vertices in the genome must be

converted into a domical structure (the phenotype). This process is accomplished by constructing the genome's convex hull, via the incremental algorithm. Once the convex hull is constructed, its edges become the structural members of the dome. Having built the dome, structural analysis is used to determine whether it performs within the constraints specified above, if not the individual is penalized using a quadratic penalty function (Richardson et al., 1989). A penalty function reduces an individual's fitness by an amount proportional to the constraint violation. It must also consider invalid structures that cannot sustain the user defined loads (because, due to the way the phenotype is constructed, there is no guarantee that they will be included in the final dome as they may not form part of the convex hull). If this situation occurs the individual is heavily penalised because the whole purpose of the structure is to support the user defined loads. Finally the dome's surface area and volume ratio is determined along with its overall weight. At the end of this process an individual's genotype is converted into its phenotype allowing its fitness to be calculated via equation (21).

$$Fitness = weight + \left(\frac{SurfaceArea}{Volume}\right) + StructuralObjective \qquad (21)$$

A sensitivity analysis showed that the weightings of the components within eqn.5 are not significant within a reasonable range around the chosen values. It is recognized that the fitness function could be more sophisticated but the main thrust of this work is to establish the representation. The parameters used for the Genetic Algorithm are:

Population size = 500, generations = 25, probability of reproduction 0.1, Probability of mutation (per genome)= 0.4, N point crossover, probability of crossover 0.5, Tournament selection (size =3) with Elitism.

Point, shuffle, addition and deletion mutation operators were all used with the choice being random. The fitness function is based on the minimisation of the enclosed volume and surface area subject to constraints relating to member buckling.

Currently this work assumes all structural members have the same cross sectional area. The genome used does not consider individual members, as an explicit parameter therefore there is no easy way of storing individual cross-sectional areas for exchange during the evolutionary process. Geodesic domes aim to have homogeneity with regard to member sizes, so this is not such a major issue. An example of the type of structure produced is given in Figure 38. As can be seen this is not a pure geodesic dome because the member lengths and the angles vary but as was stated at the start of the paper, our aim was only to produce viable and efficient domical structures.

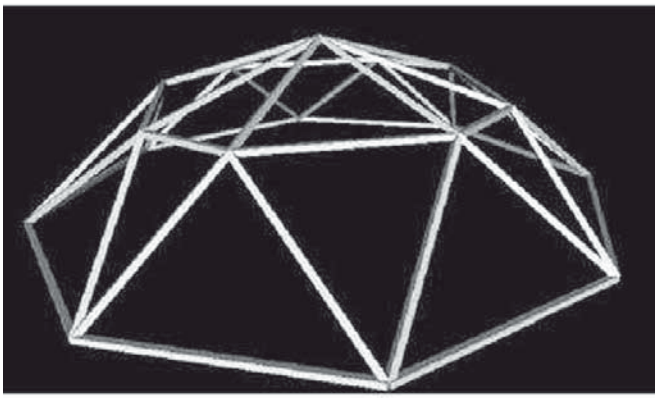


Figure 38. Example of a Dome Designed by the System

4.9 Current Research Work

As can be seen from the above, there is a problem with topological reasoning, in so far as there is no one technique that can be universally applied to all classes of problems. Instead, there are a number of discrete techniques, each of which can only be applied to a small range of topological search challenges. There are also other topological reasoning/representation techniques which have not been discussed above and likewise, these also are only applicable to certain classes of problem.

This lack of a general technique poses problems for designers who wish to look for good solutions for more complex problems. If the form of the ”best” solution is unknown, then it is impossible at the start to choose a suitable representation. Also if the form of the solution may involve several possible forms then the challenge becomes even more complex. Take for example the task of designing a bridge. Bridges can take several forms but to put it as simply as possible, the solution can be a beam (which in turn can be a truss, a box girder or simple beam and slab construction), a compression structure such as an arch or a tension structure such as a suspension or cable stayed bridge. Getting the topology of such a structure right is a major challenge and the savings and benefits can be substantial. However, there is no computationally based form of topological reasoning that could even begin to handle all these different structural forms.

There is also the question of complexity. With the current forms of representation that are used, it is only possible to represent relatively simple structural forms

within a genetic algorithm. With more complex shapes, the chromosomes become so huge and unwieldy that they present handling difficulties within the algorithm.

At present there is no clear solution to these problems but work in recent years has been looked at what are known as generative representations and also generative geometries. Space and time precludes a discussion of these techniques here but they are active research areas which seem to offer a solution to some of the challenges described above and with further research may become the solution to all of them. For the reader who wishes to know more a good starting point for generative representations is Hornby (2003) and for generative geometries, Leyton (2001). Progress to date in implementing these approaches is described in Miles et al. (2007).

Bibliography

Borkowski, A. and Grabska, E. (1995). Representing designs by composition graphs. In Smith, I., editor, *Knowledge Support Systems in Civil Engineering*, pages 27–36. IABSE, Zurich.

Bradshaw, J. (1996). *Deriving strategies for reducing pollution damage using a genetic algorithm*. Ph.D. thesis, Cardiff School of Engineering, Cardiff University.

Bradshaw, J. and Miles, J. (1997). Using standard fitnesses with genetic algorithms. *Advances in Engineering Software*.

Bucciarelli, L. and Sthapit, A. (accessed 2003). Trussworks 3d matrix truss analysis. web site 1, MIT.

de Berg, M. (2000). *Computational geometry: algorithms and applications*. Springer-Verlag, 1st edition.

Deb, K. and Gulati, S. (2001). Design of truss structures for minimum weight using genetic algorithms. *Finite Element Analysis and Design*.

Gere, J. and Timoshenko, S. (1997). *Mechanics of materials*. PWS publishing Company, 4th edition.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1st edition.

Grierson, D. and Khajehpour, S. (2002). Method for conceptual design applied to office buildings. *Journal of computing in civil engineering*.

Griffiths, D. and Miles, J. (2003). Determining the optimal cross section of beams. *Advanced Engineering Informatics*.

Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, 1st edition.

Hooper, J. (1993). *A knowledge-based system for strategic sludge disposal planning*. Ph.D. thesis, Cardiff School of Engineering, Cardiff University.

Hornby, G. (2003). *Generative representations for evolutionary design automation*. Ph.D. thesis, Brandeis University.

Khajehpour, S. and Grierson, D. (1999). Filtering of pareto optimal trade-off surfaces for building conceptual design. In Topping, B., editor, *Optimization and control in civil and structural engineering*, pages 63–70. Saxe-Coburg Press, Edinburgh.

Khajehpour, S. and Grierson, D. (2003). Profitability versus safety of high-rise office buildings. *Journal of structural and multidisciplinary optimization*.

Leyton, M. (2001). *A generative theory of shape*. Springer-Verlag, 1st edition.

Miles, J., Kwan, A., Wang, K. and Zhang, Y. (2007). Searching for good topological solutions using evolutionary algorithms. In Topping, B., editor, *Civil engineering computations: Tools and Techniques*, pages 149–172. Saxe-Coburg Press, Edinburgh.

Motro, R. (1994). Review of the development of geodesic domes. In Makowski, Z., editor, *Analysis, design and construction of braced domes*, pages 387–412. Cambridge University Press, Cambridge.

Parmee, I. (1998). Evolutionary and adaptive strategies for efficient search across whole system engineering design hierarchies. *AIEDAM*.

Parmee, I. (2001). *Evolutionary and adaptive computing in engineering design*. Springer-Verlag, 1st edition.

Richardson, J., Palmer, M., Liepens, G. and Hillyard, M. (1989). Some guidelines for genetic algorithms with penalty functions. In Anon, editor, *Proc. 3rd Int. Conf. on genetic algorithms*, pages 191–197. Morgan Kaufman, USA.

Rourke, J. O. (1998). *Computational geometry in C*. Cambridge University Press, 2nd edition.

Shamos, M. (1978). *Computational Geometry*. Ph.D. thesis, Yale University.

Shaw, D., Miles, J. and Gray, W. (2005a). Conceptual design of geodesic domes. In Topping, B., editor, *Proc. 8th Int Conf on the Application of AI to Civil, Structural and Environmental Engineering*. Saxe-Coburg, Edinburgh UK.

Shaw, D., Miles, J. and Gray, W. (2005b). Conceptual design of orthogonal commercial buildings. In Topping, B., editor, *Proc. 8th Int Conf on the Application of AI to Civil, Structural and Environmental Engineering*. Saxe-Coburg, Edinburgh UK.

Sisk, G. (1999). *The use of a GA-Based DSS for realistically constrained building design*. Ph.D. thesis, Cardiff University.

Weisstein, E. (accessed 2005). Circle point picking - mathworld - a wolfram web resource. web site 1, mathworld.wolfram.com/CirclePointPicking.html.

Wolpert, D. and MacReady, W. (1997). No free lunch theorems for optimization. *IEEE Trans. on Evolutionary Computing*.

Yang, Y. (2000). *Genetic programming for structural optimization*. Ph.D. thesis, Nanyang Technological University.

Zhang, Y. and Miles, J. (2004). Representing the problem domain in stochastic search algorithms. In Schnellenback-Held, M. and Hartmann, M., editors, *Next generation intelligent systems in engineering*, pages 156–168. EG-ICE, Essen.

CHAPTER 2

Evolutionary and Immune Computations in Optimal Design and Inverse Problems*

Tadeusz Burczyński

Department for Strength of Materials and Computational Mechanics,
Division of Intelligent Computing Systems,
Silesian University of Technology, Gliwice, Poland
e-mail: tb@polsl.pl

Institute of Computer Modelling, Cracow University of Technology,
Division of Artificial Intelligence
Cracow University of Technology, Kraków, Poland
e-mail: tburczyn@pk.edu.pl

Abstract. This chapter is devoted to applications of biologically inspired methods as evolutionary algorithms and artificial immune systems to optimization and inverse problems, related to size, shape, topology and material optimization and defect identification of structures.

1 Introduction

Evolutionary algorithms (Arabas, 2001), (Michalewicz, 1996) are methods which search the space of solutions basing on the analogy to the biological evolution of species. Like in biology, the term of an individual is used, and it represents a single solution. Evolutionary algorithms operate on populations of individuals which can be considered as a set of problem solutions. An individual consists of chromosomes. Usually it is assumed that the individual has one chromosome. Chromosomes consist of genes which play the role of design variables in optimization problems. The adaptation of the individual is computed using a fitness function. All genes of the chromosome decide about the fitness function value. A flowchart of the evolutionary algorithm is presented in Figure 1.

In the first step a initial population of individuals is created. Usually, the values of the genes of particular individuals are randomly generated. In the next step the individuals' fitness function value is computed. Then, evolutionary operators change genes of the parent population individuals, individuals are

* *This work is a result of cooperation with W.Beluch, A.Dźwigosz, W.Kuś, P.Orantek, A.Poteralski and M.Szczepanik*

selected for the offspring population, which becomes a parent population and the algorithm works iteratively till the end of the computation. The termination condition of computations can be formulated in different ways, e.g. as the maximum number of iterations.

In evolutionary algorithms the floating-point representation is applied, which means that genes included in chromosomes are real numbers. Usually, the variation of the gene value is limited.

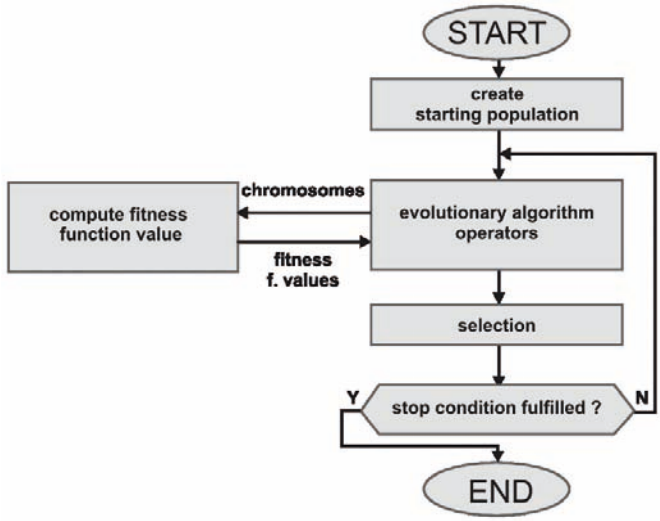


Figure 1. A flowchart of an evolutionary algorithm

A single-chromosome individual (called a chromosome) ch_i, $i=1,2,...,N$, where N is the population size, may be presented by means of a column or a row matrix, whose elements are represented by genes g_{ij}, $j=1,2,...,n$, n-the number of genes in a chromosome. The sample chromosome is presented in Figure 2.

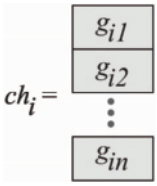


Figure 2. The structure of an individual

Evolutionary operators change gene value like the biological mechanisms of a mutation and a crossing. Different kinds of operators are presented in publications, and the basic ones are:

- an uniform mutation,
- a Gaussian mutation,
- a boundary mutation,
- a simple crossover,
- an arithmetical crossover.

The uniform mutation changes values of randomly chosen genes in a randomly selected individual. The new values of the genes are drawn in such a way that they could fulfil constrains imposed on the variation of the gene values. The diagram of how an operator works is presented in Figure 3.

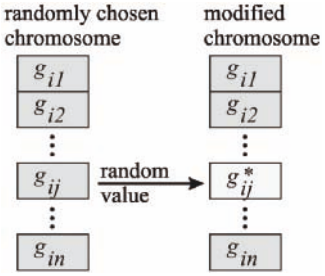


Figure 3. A diagram of an uniform mutation

The Gaussian mutation is an operator changing the values of an individual's genes randomly, similarly to the uniform mutation. New values of the genes are created by means of random numbers with the Gaussian distribution. The operator searches the individual's surrounding.

The boundary mutation (Figure 4) operates similarly to the uniform mutation, however, new values of the genes are equal to the left or right values from the gene variation range (left or right constraint imposed on gene values).

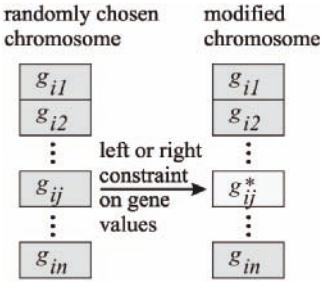


Figure 4. A diagram of boundary mutation

The simple crossover is an operator creating an offspring on the basis of two parent individuals. A cutting position is drawn (Figure 5), and a new individual consists of the genes coming partly form the first and partly form the second individual.

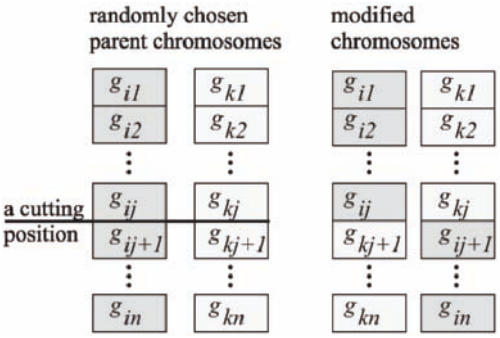


Figure 5. A diagram of a simple crossover

The arithmetical crossover has no biological counterpart. A new individual is formed similarly to a simple crossover, on the basis of two parent individuals, however, the values of the individual's genes are defined as the average value of the parent individuals' genes (Figure 6).

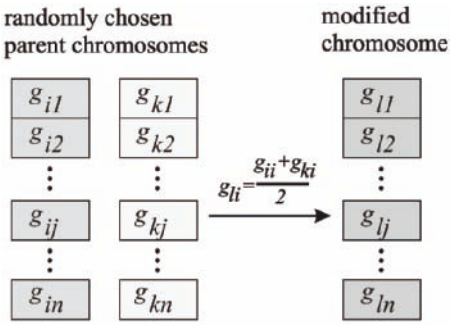


Figure 6. A diagram of an arithmetical crossover

Very important element of the evolutionary algorithm is the mechanism of selection. The probability of the individual's survival depends on the value of the fitness function. A ranking selection is performed in a few steps. First, the individuals are classified according to the value of the fitness function, then a

rank value is attributed to each individual. It depends on the individual's number and the rank function. The best individuals obtain the highest rank value, the worst obtain the lowest one. In the final step individuals for the offspring generation are drawn, but the probability of drawing particular individuals is closely related to their rank value.

2 Parallel and distributed evolutionary algorithms

2.1 Introduction

The sequential evolutionary algorithms are well known tools for global optimization (Arabas, 2001), (Michalewicz, 1996). The number of fitness function evaluations during optimization is equal to thousands or even hundred of thousands. The fitness function evaluation for the real engineering problems takes a lot of time (from seconds to hours). The long time computations can be shorten when the parallel or distributed evolutionary is used. The fitness function evaluation is done in parallel way when the parallel evolutionary algorithms are used. The distributed evolutionary algorithms operate on many subpopulations. The parallelization of the distributed evolutionary algorithm leads to two cases: first when each subpopulation uses different processor, second when the different processors can be used by each chromosome of the subpopulations.

2.2 The parallel evolutionary algorithm

The parallel evolutionary algorithms (Cantu, 1998) perform an evolutionary process in the same manner as the sequential evolutionary algorithm. The difference is in a fitness function evaluation. The parallel evolutionary algorithm evaluates fitness function values in the parallel way. Theoretically, maximum reduction of time needed to solve the optimization problem using parallel evolutionary algorithms is equal to number of used processing units. The maximum number of processing units which can be used is constrained by a number of chromosomes in the population. The flowchart of the parallel evolutionary algorithm is shown in Figure 7. The starting population of chromosomes is created randomly. The evolutionary operators change chromosomes and the fitness function value for each chromosome is computed. The server/master transfers chromosomes to clients/workers. The workers compute the fitness function and send it to server. The workers operate on different processing units. The selection is performed after computing the fitness function value for each chromosome. The selection decides which chromosomes will be in the new population. The selection is done randomly, but the fitter chromosomes have the bigger probability to be in the new population. The next iteration is performed if the stop condition is not fulfilled.

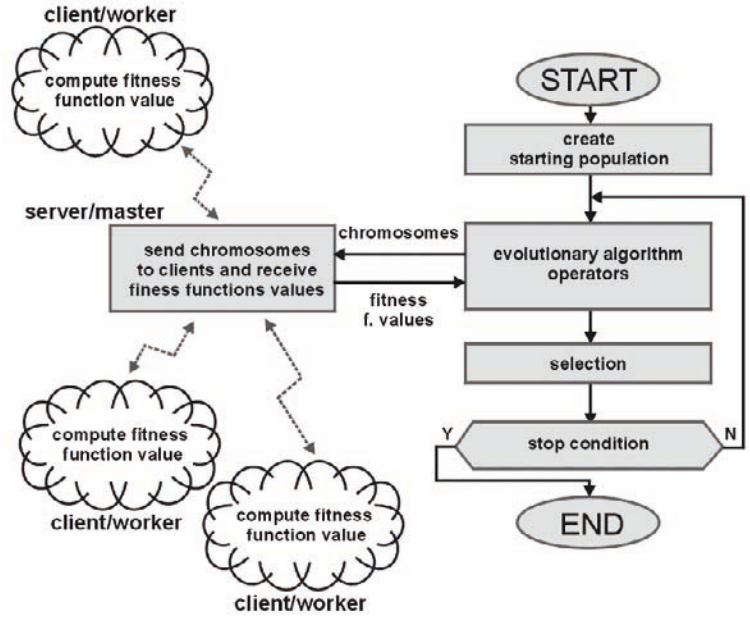


Figure 7. Parallel evolutionary algorithm

2.3 The distributed evolutionary algorithm

The distributed genetic algorithms (Aleander, 2000), (Tanese, 1989) and the distributed evolutionary algorithms (DEA) work similarly to many evolutionary algorithms operating on subpopulations. The evolutionary algorithms exchange chromosomes during a migration phase between subpopulations. The flowchart of DEA is presented in Figure 8.

When DEA is used the number of fitness function evaluations can be lower in comparison with sequential and parallel evolutionary algorithms. DEA works usually in the parallel manner. Each of the evolutionary algorithms in DEA work on a different processing unit. The theoretical reduction of time could be bigger then the number of processing units. The starting subpopulation of chromosomes is created randomly. The evolutionary operators change chromosomes and the fitness function value for each chromosome is computed.

The migration exchanges a part of chromosomes between subpopulations. The selection decides which chromosomes will be in the new population. The selection is done randomly, but the fitter chromosomes have bigger probability to be in the new population. The selection is performed on chromosomes changed by operators and immigrants. The next iteration is performed if the stop condition is not fulfilled.

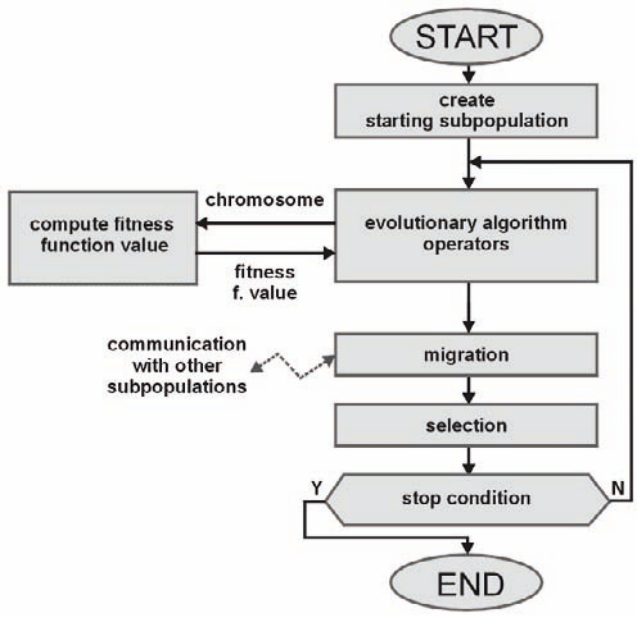


Figure 8. The distributed evolutionary algorithm (one subpopulation)

2.4 The improved distributed evolutionary algorithm

To improve scalability of the distributed evolutionary algorithm, mechanisms from the parallel evolutionary algorithm can be used. The simplest improvement is computing fitness function values in a parallel way. The maximum number of processing units which can be used is equal to a sum of chromosomes in subpopulations instead of the number of subpopulations. The flowchart of the modified distributed evolutionary algorithm is presented in Figure 9.

3 Geometry Modeling

The choice of the geometry modeling method and selection of design variables have an important influence on the final solution of the optimization problem. There is a lot of methods for geometry modeling. In the proposed approach the Bezier curves are used to model geometry of structures. This type of the curve is a superset of the more commonly known NURBS - Non-Uniform Rational B-Spline (Piegl and Tiller, 1997). Using these curves in optimization allows to reduce a number of design parameters. Manipulating by means of control points

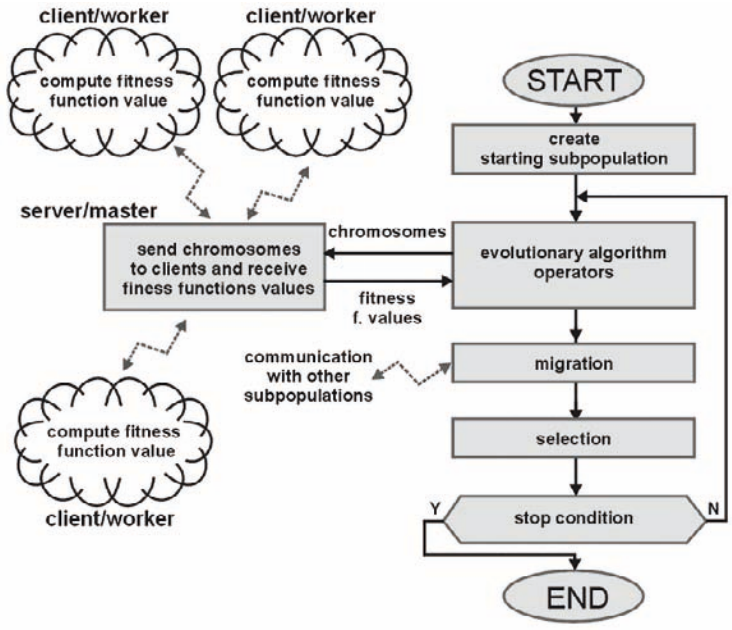


Figure 9. Improved distributed evolutionary algorithm

provides the flexibility to design a large variety of shapes. The main advantages of NURBS curves are:

- one mathematical form for standard analytical shapes as well as for free form shapes,
- flexibility to design a large variety of shapes,
- fast evaluation by numerically stable and accurate algorithms,
- invariance under transformations.

3.1 Bezier curves

An nth-degree Bezier curve is defined by:

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i \tag{1}$$

where t is a coordinate with changes range <0,1>, P_i are control points.
The basis functions $B_{i,n}$ are given by:

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-1} \tag{2}$$

The 4-th degree Bezier curve is described by following equation:

$$C(u) = (1-u)^4 P_0 + 4u(1-u)^3 P_1 + 6u^2(1-u)^2 P_2 + 4u^3(1-u)P_3 + u^4 P_4 \quad (3)$$

Example of the cubic Bezier curves is shown in Figure 10a. By changing positions of control points, one can obtain a large variety of shapes.

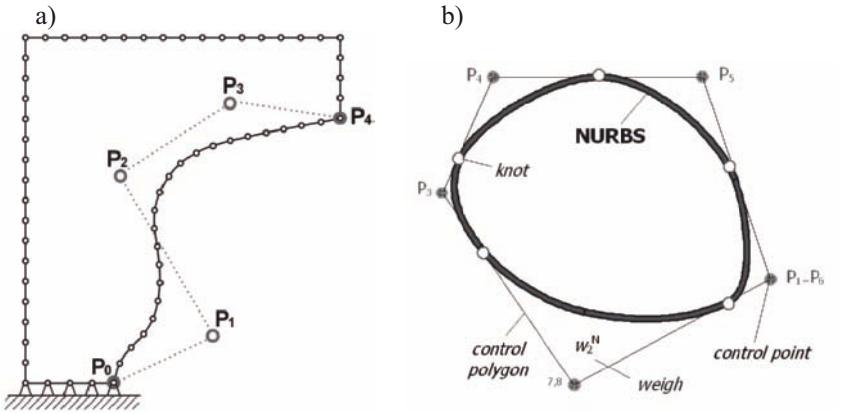


Figure 10. Example modeling:
a) 4th-degree Bezier curve, b) closed NURBS curve

3.2 NURBS curves

A NURBS curve is defined as:

$$\mathbf{C}(t) = \frac{\sum_{j=0}^{r} N_{j,n}(t) w_j \mathbf{P}_j}{\sum_{k=0}^{r} N_{k,n}(t) w_k}, \quad a \le t \le b \quad (4)$$

where: $\mathbf{P}_j$ – control points, w_j – weight of control points,
$N_{j,n}$ – nth-degree B-spline basis functions defined on the knot vector:

$$T = \left\{ \underbrace{a,\ldots,a}_{n+1}, t_{n+1},\ldots,t_{m-n-1}, \underbrace{b,\ldots,b}_{n+1} \right\} \quad (5)$$

Changing position of control points and weight of control points, it is possible to manipulate the curve precisely. A very important – from the practical point of view – feature of NURBS curves is local approximation property. It means that if the control point P_j is moved and/or the weight w_j is changed, that only a part of the curve on the interval $t \in [t_i, t_{i+p+1}]$ is modified. An example of a NURBS curve is presented in Figure 10b.

In the case of 3-D structures the boundaries as the NURBS surfaces (Figure 11) are modeled. Due to using the NURBS curves and surfaces, the number of optimized parameters can be decreased.

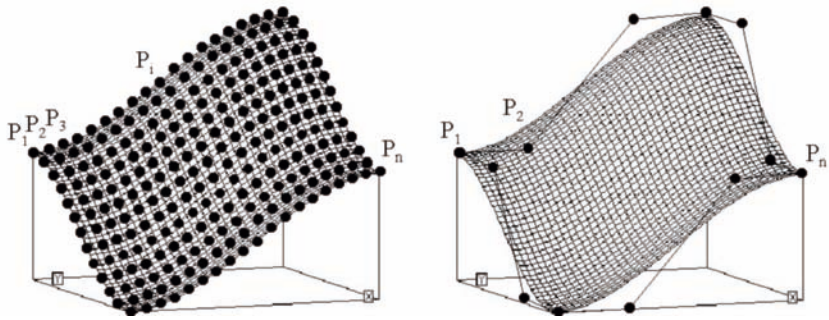


Figure 11. The modeling the boundary using the NURBS surface

Shapes of voids in 2-D structure are modelled as the: (i) – circular, (ii) – elliptical or (iii) – arbitrary, using the closed NURBS curve (Figure 12). In the case of 3-D structure shapes of voids are modeled as: (i) – spherical, (ii) – ellipsoidal or (iii) – arbitrary using the closed NURBS surface (Figure 13). Coordinates of control points $\mathbf{P}_j$ and parameters of typical geometrical figures play the role of genes in chromosomes.

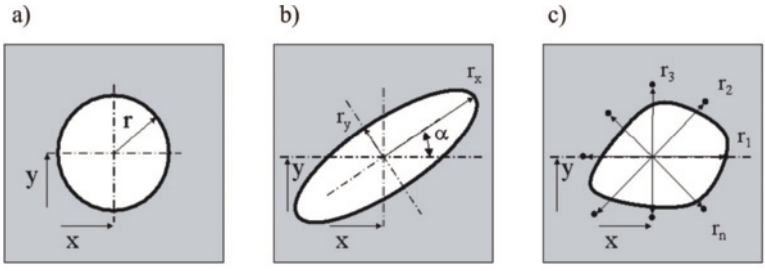


Figure 12. The modeled forms of the voids (2-D)

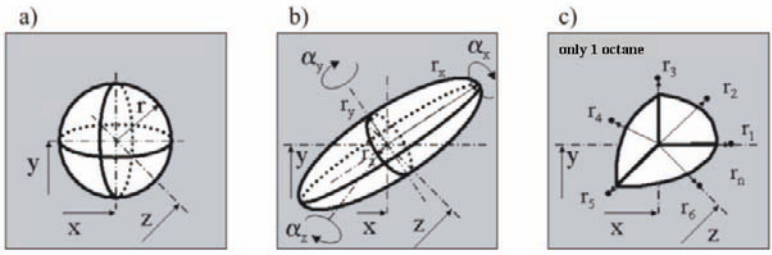


Figure 13. The modeled forms of the voids (3-D)

4 The evolutionary computations in optimization of structures under dynamical loading

4.1 Introduction

The application of classic optimization algorithms in dynamical structures is restricted by limitations referring to the continuity of the objective function, the gradient or hessian of the objective function and the substantial probability of getting a local optimum. Therefore new optimization methods, free from limitations mentioned above, have been still looked for. Those methods are known as genetic algorithms, evolutionary programming, evolutionary strategies etc. Many of them turn out to be the alternative methods of optimization for classic methods such as e.g. well known gradient methods. Particularly, the genetic algorithms are often used in solving optimization problems. Those algorithms are widely applied in many fields for solving search and optimization problem.

This chapter concerns in the application of evolutionary algorithms in the shape and topology optimization of structures being under dynamical loading.

4.2 Formulation of the optimization problem

Consider an elastic body which occupies the domain Ω bounded by boundary Γ (Figure 14). Governing equations describing an elastodynamic problem have the following form:

$$\mu u_{i,jj}(\mathbf{x},t) + (\lambda+\mu) u_{j,ji}(\mathbf{x},t) + b_i(\mathbf{x},t) = \rho \ddot{u}_i(\mathbf{x},t), \quad \mathbf{x} \in \Omega, t \in [0,t_f] \tag{6}$$

where $u_i(\mathbf{x},t)$ is a field of displacements, $b_i(\mathbf{x},t)$ is a field of body forces, λ and μ are Lame constants and ρ is a mass density.

Boundary conditions are prescribed in the following form:

$$u_i(\mathbf{x},t) = u_i^o(\mathbf{x},t), \quad \mathbf{x} \in \Gamma_u, t \in [0,t_f]$$
$$p_i(\mathbf{x},t) = \sigma_{ij}(\mathbf{x},t)n_j(\mathbf{x}) = p_i^o(\mathbf{x},t), \quad \mathbf{x} \in \Gamma_p, t \in [0,t_f] \tag{7}$$

where p_i is the field of tractions, n_j denotes the component of the outward normal $\mathbf{n}$ and $\Gamma = \Gamma_u \cup \Gamma_p$.

Initial conditions describe the field of displacements and velocities in the time *t=0:*

$$u_i(\mathbf{x},0) = d_i(\mathbf{x}), \quad \dot{u}_i(\mathbf{x},0) = v_i(\mathbf{x}), \quad \mathbf{x} \in \Omega \tag{8}$$

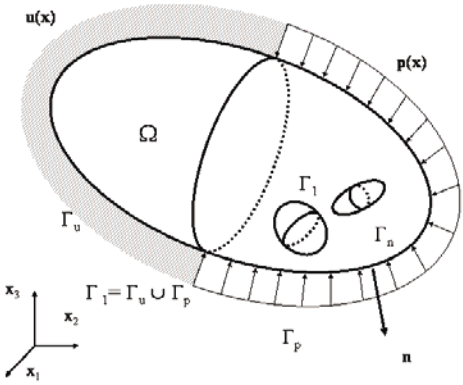


Figure 14. The elastic body

The problem of the shape and topology optimization of elastic structures being under dynamical loads can be formulated as the minimization of the volume of the structure

$$\min J \equiv \int_{\Omega} d\Omega \tag{9}$$

subjected to the constraints imposed on equivalent stresses and displacements

$$\sigma_{eq}(\boldsymbol{x},t) - \sigma_o \le 0$$
$$u(\boldsymbol{x},t) - u_o \le 0 \tag{10}$$

where: $u = \sqrt{u_i u_i}$ $\mathrm{x} \in \Gamma$ or $\mathbf{x} \in \Omega$, $\mathrm{t} \in \mathrm{T} = [0,t_f]$, σ_o and u_o are admissible equivalent stresses and displacement, respectively.

There is also the alternatively formulation of shape and topology optimization in which one minimizes a functional

$$J \equiv \int_T \int_\Omega \Psi(\sigma, \varepsilon, u) d\Omega dt + \int_T \int_\Gamma \Phi(u, p) d\Gamma dt \tag{11}$$

with the constraints imposed on the volume of the structure

$$J \equiv \int_\Omega d\Omega - Vo \leq 0 \tag{12}$$

Integrands Ψ and Φ are the arbitrary functions of their arguments. Using the evolutionary algorithms, the minimization of (9) and (11) is performed with respect to shape design variables. In order to evaluate the functional (11) and constraints (10) the boundary element method was applied (Burczyński, 1995).

4.3 Example of evolutionary shape optimization under dynamical loads

The example concerns minimization of the volume of a support (Figure 15). The support is loaded by dynamical loading $F(t)=F_0\sin(\omega t)$, where F_0=10kN (Figure 16). The optimization fitness function (9) was used. The constraints on the values of the displacements were imposed.

The surface of the support was modeled using the NURBS surface. The coordinates of the marked points (Figure 17) (control points of the NURBS surface) were modified.

The following parameters of the evolutionary algorithms were applied: *pop_size*: 50, *max_life*: 400. The optimal structure was shown in Figure 18.

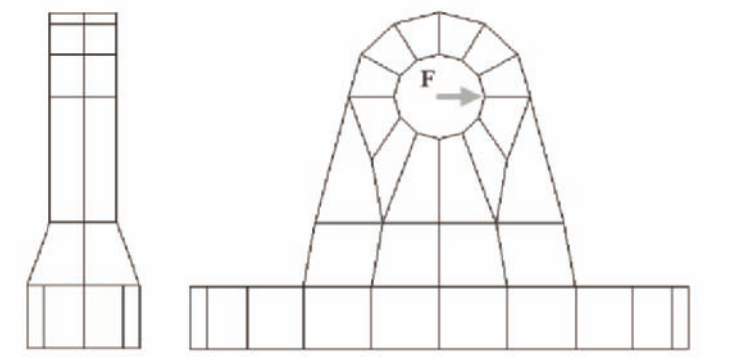


Figure 15. The support

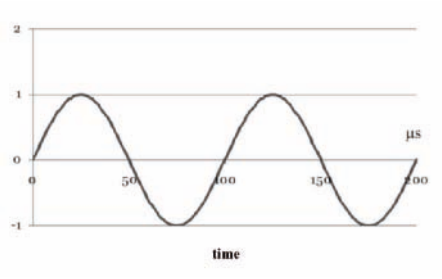


Figure 16. The forced function

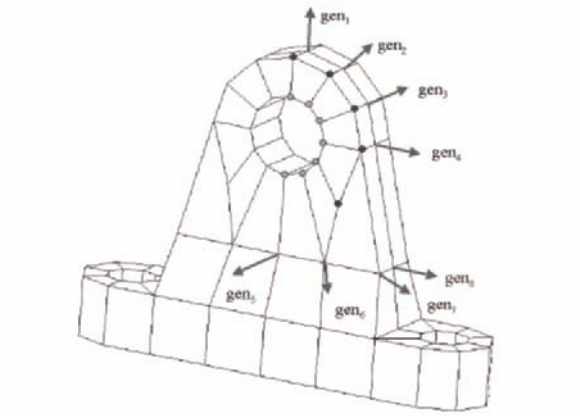


Figure 17. The location of the control points of the NURBS surface

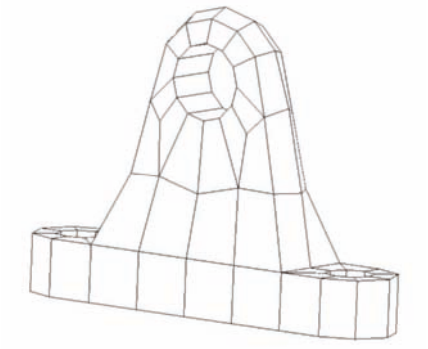


Figure 18. The support after optimization

5 Evolutionary Optimization and Identification in Thermomechanical Problems

5.1 Introduction

This chapter deals with a coupling of the distributed evolutionary algorithms and the boundary element method (BEM) in optimization of structures being in thermomechacal conditions. The problem is considered in the framework of uncoupled steady state thermoelasticity.

Evolutionary algorithms (EA) are well known tool for solving of optimization problems. The drawback of EA is long time needed for computing (Arabas, 2001), (Michalewicz, 1996). This work is an extension of previous papers in which coupling EA and BEM has been used in shape optimization and identification of thermoelastic structures (Burczyński and Długosz, 2002), (Długosz, 2001, 2004). The distributed evolutionary algorithms (Tanese, 1989), (Cantû-Paz, 1998, 1999, 2000) can shorten computing time.

In order to reduce the number of design variables one employs parametric curves, like Bézier curves, B-Splines and NURBS, which allow modelling complicated shapes with a relatively small number of control points.

In this section the boundary element method and the distributed evolutionary algorithm are employed to solve shape optimization and identification problems. Bezier and NURBS parametric curves are used to model the geometry of structural elements.

5.2 Governing equations and objective functions

Governing equations for an uncoupled thermoelastic problem have the following form:

$$kT_{,ii}(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega \tag{13}$$

$$\mu u_{i,jj}(\mathbf{x}) + (\lambda + \mu) u_{j,ji}(\mathbf{x}) + b_i(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega \tag{14}$$

where body forces are given as follows:

$$b_i = -\frac{2\mu(1+\nu)}{1-2\nu} \alpha T_{,i} \tag{15}$$

where k is a thermal conductivity, α stands for a coefficient of thermal expansion and ν is a Poisson's ration.

Boundary conditions are prescribed in the following form:

$$\begin{aligned} &u_i(\mathbf{x},t) = u_i^o(\mathbf{x},t), \quad \mathbf{x} \in \Gamma_u, \\ &p_i(\mathbf{x},t) = \sigma_{ij}(\mathbf{x},t) n_j(\mathbf{x}) = p_i^o(\mathbf{x},t), \quad \mathbf{x} \in \Gamma_p, \end{aligned} \tag{16}$$

where $\Gamma = \Gamma_u \cup \Gamma_p$ and

$$\begin{aligned} &T(\mathbf{x}) = T^o(\mathbf{x}), \quad \mathbf{x} \in \Gamma_T \\ &q(\mathbf{x}) - kT_{,i} n_i = q^o(\mathbf{x}), \quad \mathbf{x} \in \Gamma_q \\ &q(\mathbf{x}) = h(\mathbf{x})[T(\mathbf{x}) - T_f(\mathbf{x})], \quad \mathbf{x} \in \Gamma_h \end{aligned} \tag{17}$$

where q is the conductive heat flux, h is heat transfer coefficient, T_f is core temperature of the fluid, $\Gamma = \Gamma_T \cup \Gamma_q \cup \Gamma_h$.

Equations (13) and (14) with boundary conditions (16) and (17) are solved by means of the boundary element method (Białecki *et al*, 2005).

The following criterion is proposed for evolutionary optimization of structures being under thermomechanical loading:

$$J_0 = \int_\Gamma \left(\frac{u}{u_0} \right)^{2n} \mathrm{d}\Gamma \tag{18}$$

where u is a field of boundary displacements, u_0 is a reference displacement, n is natural number. Minimization of the functional (18) reduces displacements on the selected part of the boundary.

In inverse problems in order to find an unknown inner boundary the measured values of displacements and temperatures in the sensor points on the external boundary were used. Measured values are simulated numerically by solving the boundary value problem of thermoelasticity by BEM for the actual position of internal voids. This problem is solved by minimization of the following functional:

$$J_0 = \delta \sum_{k=1}^{M} (u^k - \hat{u}^k)^2 + \eta \sum_{l=1}^{N} (T^l - \hat{T}^l)^2 \tag{19}$$

where $\hat{u}^k$ and $\hat{T}^l$ are measured values of displacements and temperatures, u_k and T_l are computed values of displacements and temperatures in boundary points k and l, respectively, δ and η are weight coefficients, M, N are numbers of sensors.

5.3 Numerical examples

The 2-D structures modelled in plain strain state are considered in identification and shape optimization problems. Table 1 contains material parameters applied for every numerical test.

Table 1. Material parameters

Shear modulus	80GPa
Poisson ratio	0.23
Coefficient of thermal exp.	$12.5 \cdot 10^{-6}$ 1/ºC

Example 1. The identification of a circular void in the rectangular plate shown in Figure 19 is considered. The fitness function given by (19) is applied. Displacements are measured in the sensor points **1** and **2** whereas temperature is measured in the sensor points **3** and **4**. The boundary was divided into 48 elements.

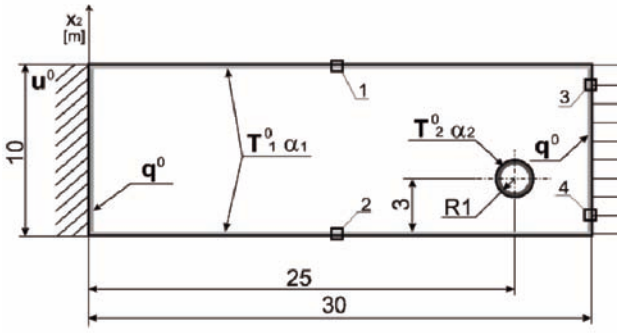


Figure 19. A rectangular plate with circular void

The position and radius of the inner boundary were searched. Table 2 contains parameters of the boundary conditions.

Table 2. Parameters of the boundary conditions

T^0_1	20°C
T^0_2	500°C
q_0	0
p_0	100kN/m
α_1	$20W/m^2K$
α_2	$1000W/m^2K$
u_0	0

The speedup s of computation can be expressed as time need to solve problem on 1 processing unit t_1 computer divided by time on n-processing units t_n:

$$s = \frac{t_1}{t_n} \tag{20}$$

The number of processing units vary from 1 to 3. The two computers with two SMP processing units are used. Table 3 contains parameters of distributed evolutionary algorithm.

Table 3. Parameters of distributed evolutionary algorithm

Number of subpopulations	2
Number of chromosomes	10
Number of genes.	3
Constrain on gene 1 (x_1 coordinate)	0,5 ÷ 29,5
Constrain on gene 2 (x_2 coordinate)	0,5 ÷ 9,5
Constrain on gene 3 (radius)	0,5 ÷ 3,0

The speedup of the improved distributed evolutionary algorithm is shown in Figure 20. The linear speedup is theoretical maximal speedup of the parallel evolutionary algorithm.

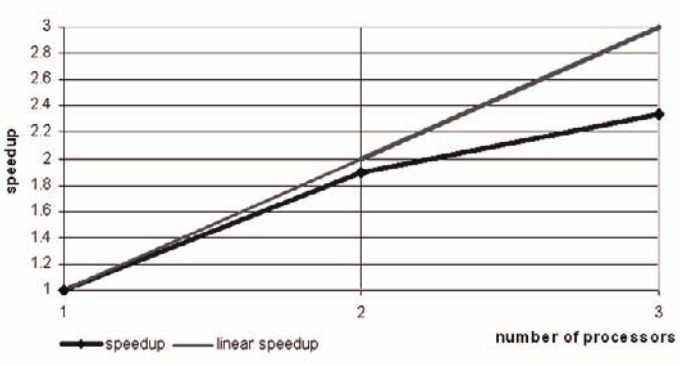


Figure 20. The speedup of improved distributed evolutionary algorithm

Table 4 contains the best result (after 8897 iterations) and relative errors for coordinates and radius of internal void.

Table 4. Results of the tests

x_1 coordinate	25.28978
x_2 coordinate	2.989090
radius	0.997610
value of the fitness function	0.000030
x_1 coordinate error	1,16%
x_2 coordinate error	0,36%
radius error	0,24%

Example 2. The structure with three inner boundaries, shown in the Figure 21, is considered.

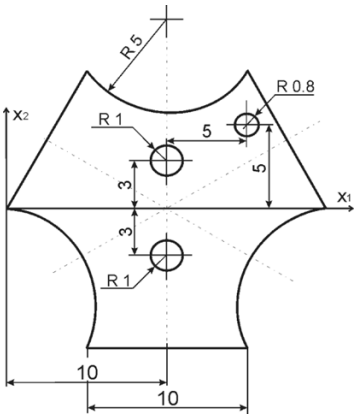


Figure 21. Structure with circular voids

In the identification problem the positions, radii and number of the voids (form 1 to 5) were searched. The fitness function given by (19) was applied. Figure 22 shows thermomechanical loading for the structure and Table 5 contains values of the boundary conditions.

Table 5. Parameters of the boundary conditions

T^0_1	100°C
T^0_2	20°C
T^0_3	0°C
p_0	100kN/m
α_1	1000W/m^{2}K
α_2	20W/m^{2}K
u_0	0

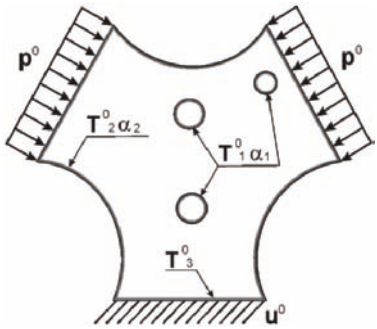


Figure 22. Boundary conditions

The external boundary was divided into 39 elements, whereas each internal boundary into 8. The boundary sensor points of displacement and temperatures are located at every node of external boundary except the part where u_0 is prescribed.

The chromosome contains 15 genes (3 for each void). The void was generated for the value of radius greater than 0.5. Table 6 contains parameters of distributed evolutionary algorithm. Figure 23 and Table 7 show results of identification.

Table 6. Parameters of distributed evolutionary algorithm

Number of subpopulations	2
Number of chromosomes	10
Constrain on x_1 coordinate	0,5 ÷ 19,5
Constrain on x_2 coordinate	-8,2 ÷ 8,2
Constrain on radius	0 ÷ 2

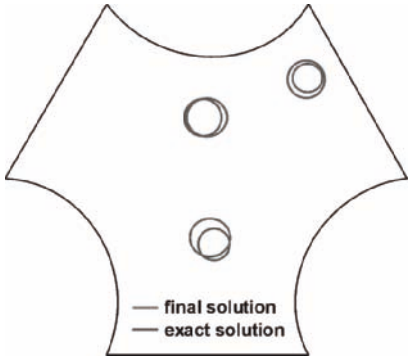


Figure 23. Results of identification

Table 7. Results of the identification

Void 1	x_1 coordinate	9.78084	2.19%
	x_2 coordinate	2.97841	0.72%
	radius	0.92832	7.17%
Void 2	x_1 coordinate	10.3143	3.14%
	x_2 coordinate	-3.24751	8.25%
	radius	0.77259	22.74%
Void 3	x_1 coordinate	14.8896	0.74%
	x_2 coordinate	4.86734	2.65%
	radius	0.93014	16.27%

Example 3. A square plate with a circular void is considered (Figure 24).

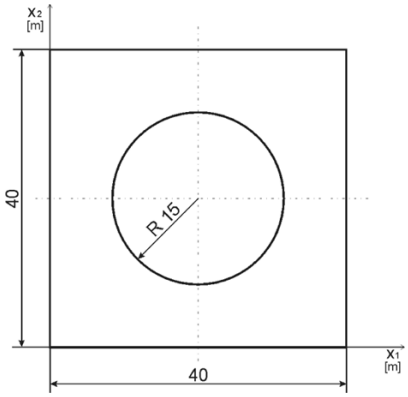


Figure 24. Square plate with circular void

For the sake of the symmetry only a quarter of the structure is taken into consideration. The considered quarter of the structure contains an internal boundary shown in the Figure 25. Prescribed values of the boundary conditions are presented in Table 8.

Table 8. Parameters of the boundary conditions

T^0_1	300°C
T^0_2	20°C
q^0	0
p_0	100kN/m
α_1	1000W/m^{2}K
α_2	20W/m^{2}K
u_0	0

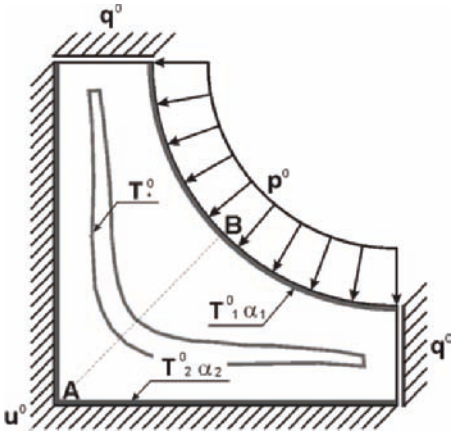


Figure 25. Boundary conditions

The model consists of 90 boundary elements. The objective of shape function is minimization of the radial displacements given by the functional (18) on the boundary where tractions p^0 are prescribed. The optimization problem consists in searching an optimal:

- shape of the internal boundary;
- width of the gap;
- distribution of the temperature T^0_* on the internal boundary.

Shape of the internal boundary was modeled using Bezier curve which consists of 7 control points, whereas width of the gap and temperature T^0_* using Bezier curve consist of 6 control points (Figure 26).

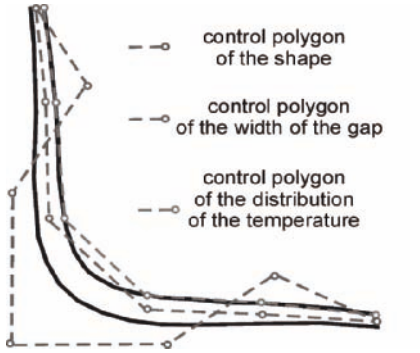


Figure 26. Modelling the shape, width of the gap and distribution of the temperature on the boundary

For the sake of the symmetry along line AB (Figure 25) the total number of design parameters was equal to 13. The range of the variability of each control point for the width of the gap is between 0.2 and 0.8, whereas for the temperature is between 5°C and 80°C.

Table 9 and Figure 27 contain final results of evolutionary optimization.

Table 9. Results of the optimization

Shape of the internal boundary	x1 coordinate of control point 1	1.1124
	x2 coordinate of control point 1	13.3259
	x1 coordinate of control point 2	2.4609
	x2 coordinate of control point 2	7.0000
	x1 coordinate of control point 3	1.6232
	x2 coordinate of control point 3	5.0000
	x1 = x2 coordinate of control point 4	-4.0853
Width of the gap	control point 1	0.4313
	control point 2	0.2752
	control point 3q	0.8000
Distribution of the temperature on the internal boundary	control point 1	48.9698
	control point 2	41.8679
	control point 3	5.0000

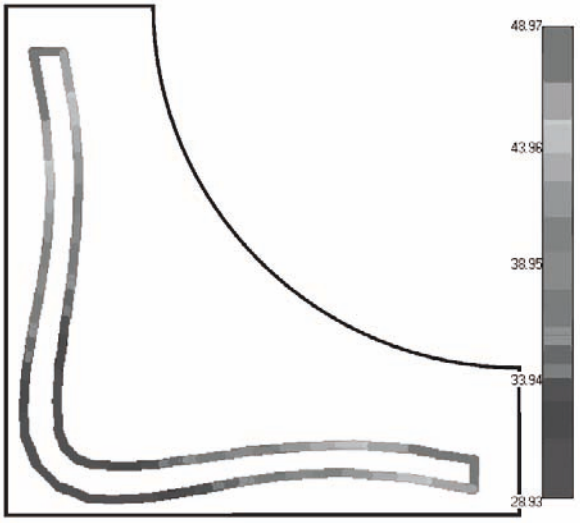


Figure 27. The optimal shape, width and distribution of the temperature on the gap

6 Distributed Evolutionary Algorithm in Optimization of Nonlinear Structures

6.1 Introduction

The shape optimization problem of elasto-plastic structures can be solved using methods based on sensitivity analysis information (Bendsoe and Sokołowski, 1988) or non gradient methods based on genetic algorithms (Burczyński and Kuś, 2001, 2003). This chapter is devoted to the method based on the parallel and distributed evolutionary algorithms. Applications of evolutionary algorithms in optimization need only information about values of an objective (fitness) function. The fitness function is calculated for each chromosome in each generation by solving the boundary – value problem of elasto-plasticity by means of the finite element method (FEM) (Zienkiewicz, 2000) or the boundary element method (BEM) (Brebbia, 1984), (Burczyński, 1995). This approach does not need information about the gradient of the fitness function and gives the great probability of finding the global optimum. The main drawback of this approach is the long time of calculations. The applications of the parallel and the distributed evolutionary algorithms can shorten the time of calculations but additional requirements are needed: a multiprocessor computer or a cluster of computers are necessary.

The chapter describes evolutionary optimization of structures with material and geometrical nonlinearities. The two types of analysis are presented, static nonlinear and time dependent – forging process optimization. The evolutionary optimization of nonlinear structures using distributed and parallel evolutionary algorithms can be found in works Burczyński and Kuś (2001, 2003, 2004), Burczyński *et al* (2004), Kuś (2002). The optimization of forging process based on gradient algorithms are presented in Badrinarayanan (1997), Zabaras *et al* (2000), Zhao *et al* (2002). The evolutionary approach was considered in António and Douardo (2002), Burczyński and Kuś (2003, 2004). The numerical examples of plate, shells and axisymmetrical structures optimizations are show.

6.2 Formulation of the evolutionary optimization

Structures made from nonlinear material with hardening

A body which occupies the domain Ω bounded by the boundary Γ is considered. The body is made from an elasto-plastic material with hardening. Boundary conditions in the form of displacements and tractions are prescribed and body forces are given. One should find the optimal shape of the body to minimize areas of the plastic zones in the domain Ω. Such a optimization criterion can be achieved by minimizing the following fitness function:

$$F = \int_{\Omega} \left(\frac{\sigma_a}{\sigma_0} \right) d\Omega \quad \text{where } \sigma_a = \begin{cases} \sigma_{eq} & \text{when } \sigma_{eq} \geq \sigma_p \\ 0 & \text{when } \sigma_{eq} < \sigma_p \end{cases} \tag{21}$$

where σ_{eq} means the Huber – von Mises equivalent stress, σ_p is the yield stress and σ_0 is the reference stress.

Shape optimization of structures with geometrical nonlinearities is performed by minimizing structure displacements. The fitness function can be formulated in the form:

$$F = \int_{\Omega} \left(\frac{u}{u_0} \right)^2 d\Omega \tag{22}$$

where u is the displacement, u_0 is the reference displacement.

Constrains in the form of admissible volume of the structure and boundary values of design variables are imposed. Shape of the optimized structure can be defined using NURBS (Non-Uniform Rational B-Spline) (Piegl and Tiller, 1997). There is a need of conversion curves into line segments and than the structure is meshed using triangle finite elements (FEM) or using boundary elements and cells (BEM). The Triangle (Shewchuk, 1996) code was used for the body meshing. Coordinates of control points of the NURBS curve play the role of genes in the chromosome.

Forging process optimization

The forging process in highly nonlinear. Three different fitness functions were used during optimization. The first one is a measure between axisymmetrical shape of the forged detail and the wanted one.

$$F = \int_{y} \Delta r(y) \, \mathrm{d}\, y \tag{23}$$

The meaning of the $\Delta r(y)$ is show in Figure 28. The optimal fitness function value is known and is equal to zero.

The MSC.Marc was used to solve the forging problem. The axisymmetrical bodies were considered. The forging process was modeled with the use of two bodies – rigid for a anvil and elastoplastic for a preform. The contact with Columb friction were used. The isothermal conditions were considered. The perform material was modeled as a viscoplastic material using equation:

$$\sigma = A\left(\varepsilon_0 + \varepsilon\right)^m + B\dot{\varepsilon}^n \tag{24}$$

where σ - is a stress, ε - strain, $\dot{\varepsilon}$ - strain velocity, ε_0 - preliminary strain, A, B, n, m - are material coefficients.

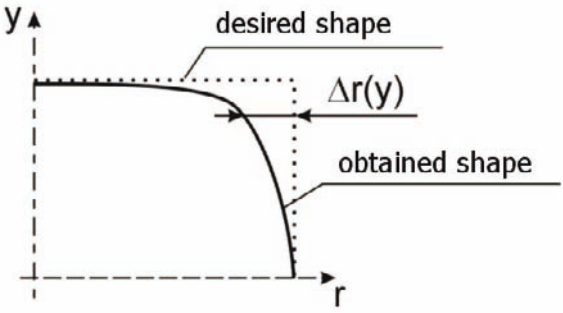


Figure 28. The obtained and wanted shape of the forged detail

The second and third fitness functions depend on plastic strains values. The idea of using these functions is to equalize plastic strains distribution in the body. The fitness function can be expressed as a double integral over the time and over the area of the structure with the difference between plastic strains ε_p and mean plastic strains ε_{av} as an integrand:

$$F = \int_o^T \int_\Omega \left(\varepsilon_p - \varepsilon_{av} \right) \, \mathrm{d}\Omega \mathrm{d}t \tag{25}$$

The third fitness function is a double integral over the time and over the area of the structure with plastic strains as an integrand:

$$F = \int_0^T \int_\Omega \varepsilon_p \, \mathrm{d}\Omega \mathrm{d}t \tag{26}$$

6.3 Numerical examples

A material with the material characteristic presented in Figure 29 is used in test problems (*Examples 4 - 6*).

Example 4. A 2-D structural element is considered (Figure 30a). The material data and parameters of the distributed evolutionary algorithm parameters are: E_1=20 GPa, E_2=0.5 GPa, σ_p=250 MPa, ν=0.3, thickness 5 mm, load value 110 N/mm, maximum body area 8000 mm^2, number of chromosomes 500, number of generations 250, number of populations 4.

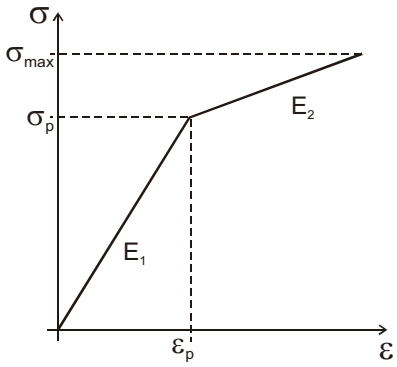


Figure 29. Uniaxial stress-strain curve for material used in tests
E_1 and E_2 are Young's moduli, ε_p is yield strain and σ_p is yield stress.

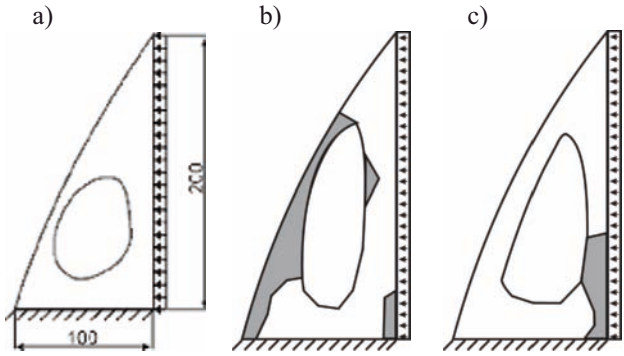


Figure 30. Optimized plate: a) geometry,
b) best after 1st generation, c) best after 196th generation

The external boundary and the hole boundary undergo shape optimization. The external boundary was modeled using the NURBS curve with 3 control points (one of them can be moved – 2 design variables) and the internal hole was modeled using 4 control points NURBS curve (each can be moved – 8 design variables). The fitness function was computed using FEM. The shape of the

boundary after first and 196 generation is shown in Figure 30b,c. The plastic areas are marked using the gray color.

In order to examine the DEA for various number of computers the computing time was measured for 15000 fitness function evaluations. Computers had AMD Duron 750 processors. The computing time versus the number of computers is given in Table 10. The number of computed fitness functions as function of the number of computers is shown in Figure 31. The starting population was the same for each test. Problem was simpler that one shown above – finite element mesh had lower number of elements.

Table 10. The computing time in function of computers number

number of computers	computing time [s]
1	745
2	374
3	258
4	195

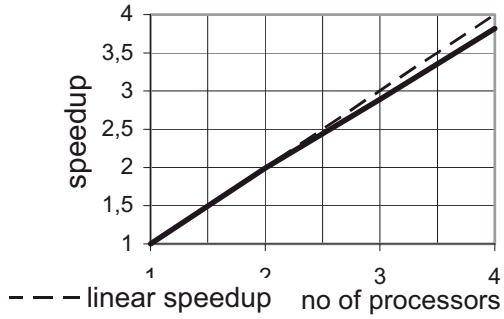


Figure 31. Speedup of computations

Example 5. The problem of shape optimization of a half K-structure is considered (Figure 32a). The material data and parameters of the DEA are: E_1=20 GPa, E_2=0.5 GPa, σ_p=150 MPa, ν=0.3, thickness 5 mm, load value 50 N/mm, maximum body area 30000 mm^2, number of chromosomes 200, number of generations 500, number of populations 4. The traction-free boundary is modeled by 2 NURBS curves with 3 control points each. The fitness function was evaluated by the BEM. The shape of the structure after first and 476 generation is shown in Figure 32b,c. The grey color was used to mark the plastic areas. Computing time – 72 minutes.

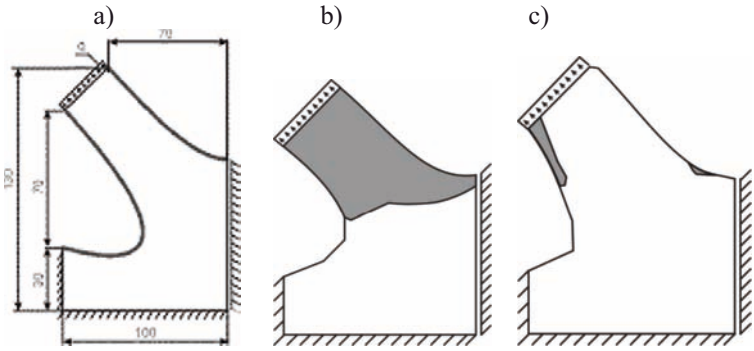


Figure 32. Half of K-structure: a) geometry, b) best after 1st generation, c) best after 476th generation

Example 6. A shell structure containing 10 holes with constant radii is considered. (Figure 33a). The optimization problem is to find optimal positions of holes for criterion of minimum an integral over shells displacements. The structure was computed considering large displacements The fitness function was evaluated using MSC.Nastran. The shape of the shell after first and 500th generation is shown in Figure 33b,c.

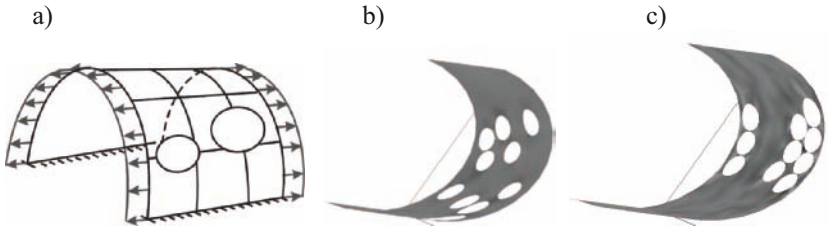


Figure 33. Shell: a) geometry, b) best after 1st generation, c) best after 500th generation

Example 7. The shape optimization problem of a perform was considered. The open die forging is simulated. The flat anvil was used. The goal of the optimization is to find shape of the perform which leads to cylindrical shape after forging. The geometrical parameters are shown in the Figure 34. The material parameters in the constitutive equation (24) for aluminum in 350 ° C were used: *A=26.478*, *B=24.943*, *m=0.1629*, *n=3.4898*. The friction coefficient was equal to *0.5*. The time step was *0.002s*, the number of steps *200*, Speer of the anvil *75mm/s*. The fitness function (23) was used during the optimization process.

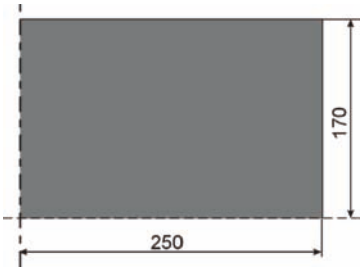


Figure 34. The desired shape of the preform after forging

The geometry of the preform (Figure 35) was modeled using NURBS curve with 4 control points. Coordinates of the control points were defined using 6 genes values (g_1-g_6).

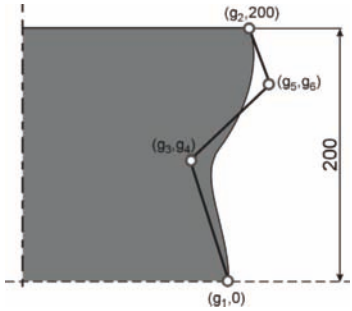


Figure 35. The geometry of the preform

The constraints imposed on the genes values are shown in Table 11.

Table 11. The constraints on the genes values

gen	minimum [mm]	maksimum [mm]
g_1	50	250
g_2	50	250
g_3	50	300
g_4	10	100
g_5	50	300
g_6	110	190

The number of chromosomes was 25, probability of the uniform mutation 25%, probability of the Gaussian mutation 62.5%, probability of the simple crossover 6.25%, probability of the arithmetic crossover 6.25%.

The best result was achieved after 638 generations (15362 fitness function computations). The best found shape of the perform is presented in the Figure 36a and the shape after forging in Figure 36b.

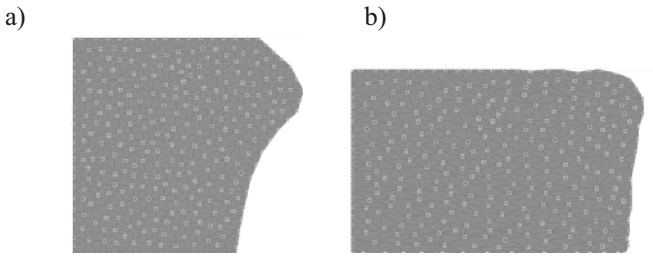


Figure 36. a) The best fund shape of the preform, b) the shape of the preform after forging

Example 8. This example describes evolutionary optimization of two stage axisymmetrical preform forging. Shape optimization of an anvil in the first stage was performed. The first stage is the open die forging, the second one is closed die forging. The criterions were expressed as (25) and (26). The results obtained for both criterions are very close to each other. The shape of the anvil described using NURBS function is shown in the Figure 37. The 8 parameters of the NURBS curve were searched.

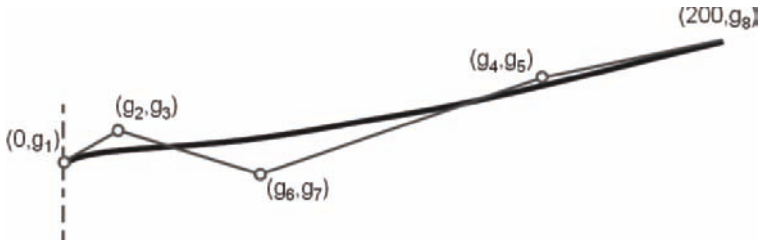


Figure 37. The shape of the anvil

The perform had cylindrical shape. The material parameters were the same as in the *Example 7*. The friction coefficient was equal to *0.3*. The model was discretized using quadrilateral elements. The evolutionary algorithm with 10 chromosomes was used. The Gaussian mutation and the simple crossover operators were applied.

The Figure 38a shows results obtained after the flat anvil forging in the first stage and Figure 38b after closed die forging in the second stage. The best found result is presented in Figure 39.

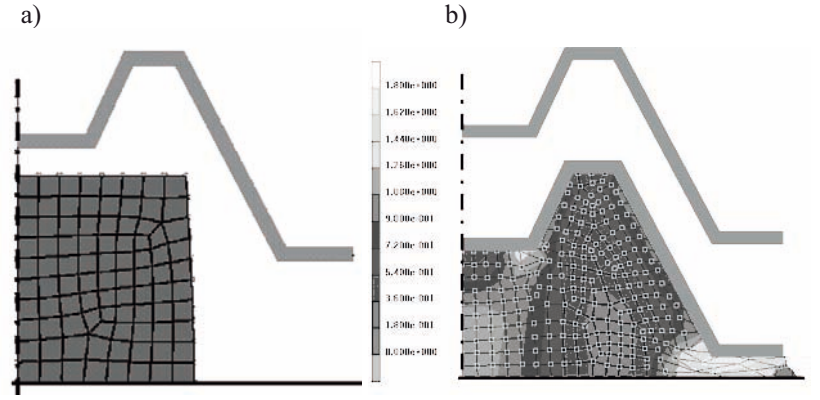


Figure 38. The shape of the preform after a) first stage, b) second stage of forging

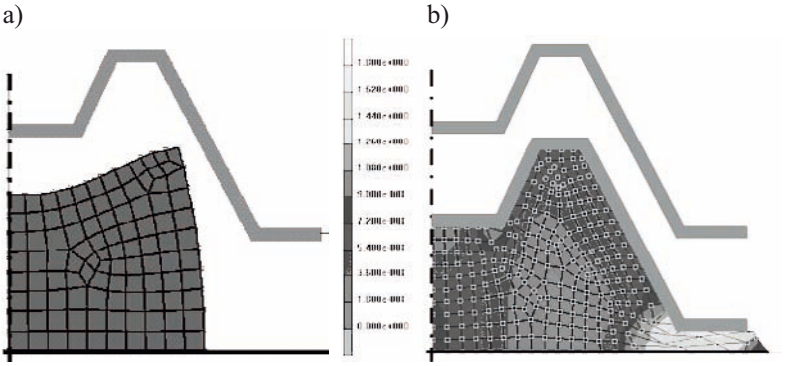


Figure 39. The shape of the preform obtained using the best anvils after a) first stage, b) second stage of forging

The speedup of computations, expressed by (20), in this case is presented in Figure 40.

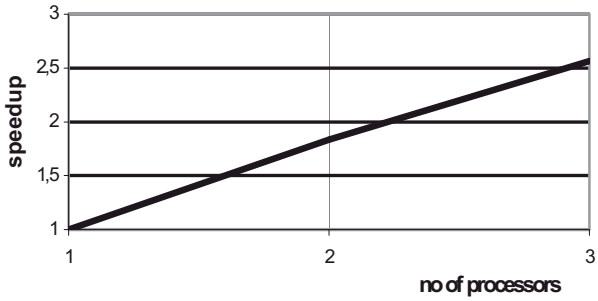


Figure 40. Speedup of computations

7 Topological Optimization of 2-D and 3D Structures Using Evolutionary Computing

7.1 Introduction

Shape and topology optimization have been active research areas for some time. Recently, several innovative approaches for topology optimization have been developed. Perhaps one of the simplest optimization method is the method based on removing inefficient material from a structure, which is named the evolutionary structural optimization (Xie and Steven, 1997), however this method is not based on application of the evolutionary algorithm but on different rejection criteria for removing material which depending on the types of design constraints.

One of the most famous of the structural optimization approaches is the approach based on material homogenization method which was introduced by Bendsøe and Kikuchi (1988) and has been applied to various optimization problems. The homogenization design method assumes introduction of the periodic microstructures of a particular shape into the finite elements of the discretized domain. The size and orientation of the microstructures in the elements determine the density and structural characteristics of the material in the elements. An optimization process consisting in application of the mathematical programming techniques leads to the minimization of the structure' compliance by changing of the orientation and size of the microstructures. In effect of the optimization process composite structures emerges.

Another approach to the structural optimization is based on generating inside a domain a new void (so-called bubble) of the basis on special criteria and next on performing simultaneous shape and topology optimization. This approach was originated by Eschenauer and Schumacher (1995). Coupling this

approach and boundary elements with genetic algorithms was considered by Burczyński and Kokot (1998). From the mathematical point of view this approach is based on replacing a one-connected domain by a multi-connected domain.

Next interesting approach assumes discretization of the domain into binary material/void elements introduced by Anagnostou *et al* (1992). This approach was developed by Kirkpatrick *et al* (1983), who proposed finding of the optimal material configuration within the design domain by use of simulated annealing. While Jensen (1992) and Sandgren *et al* (1990), proposed application of the genetic algorithm in order to solve similar optimization problems. This approach has been developed by Chapman *et al* (1995).

One of the most interesting of the recent approaches to the structural optimization problem is method named Multi-GA System introduced by Woon *et al* (2003) which assumes application of two simultaneously and parallel running genetic algorithms. The first external genetic algorithm is used to define the optimum shape of the structure through operating on the external boundary while the second internal is used to optimise the internal topology. This method does not require application of the post-processing or additional algorithms to generate smooth boundaries.

Presented in this chapter results are based on application of the evolutionary algorithm and finite element method to the optimization problems of 2-D and 3D structures. They are an extension of previous works concerning such an optimization problems (Burczyński *et al*, 2003, 2007), (Szczepanik 2003). Recently, evolutionary methods have found various applications in mechanics, especially in structural optimization (Burczyński and Osyczka, 2004). The main feature of those methods is to simulate of biological processes based on heredity principles (genetics) and the natural selection (the theory of evolution) to creating of optimal individuals (solutions) presented by single chromosomes. Evolutionary algorithms are usually applied in the situations when the optimization problems are too complicated for the traditional gradient optimization methods.

The task considered in the present work is related to such a problem. This task consists in creating an effective optimization algorithm for 2-D and 3D structures in respect of topology, shape and material or thickness arrangement. The main advantage of the evolutionary algorithm is the fact that this approach does not need any information about the gradient of the fitness function and gives a strong probability of finding the global optimum. The fitness function is calculated for each chromosome in each generation by solving the boundary-value problem by means of the finite element method (FEM). In order to solve the optimization problem the fitness function, design variables and constraints should be formulated.

7.2 Formulation of the problem

Consider a structure (plate in plane stress/strain, bending plate or shell) which, at the beginning of an evolutionary process, occupies a domain $\Omega_0 \left(in\ E^d,\ d = 2\ or\ 3 \right)$, bounded by a boundary Γ_0. The domain Ω_0 is filled by a homogeneous and isotropic material of a Young's modulus E_0 and a Poisson's ratio ν. The thickness of the structure g_0 is also constant at the beginning of the evolutionary process. The 2-D structures are considered in the framework of theory of linear elasticity. During the evolutionary process the domain Ω_t, its boundary Γ_t and the field of Young's modulus $E(\boldsymbol{x}) = E_t,\ \boldsymbol{x} \in \Omega_t$ or the thickness $g(\boldsymbol{x}) = g_t$ can change for each generation t (for t=0, E_0=const, g_0=const). The evolutionary process proceeds in an environment in which the structure fitness is describing by two possibilities:

– the objective is to minimize of the stress functional

$$J = \int_{\Omega} \psi(\sigma) d\Omega \tag{27}$$

where ψ is an arbitrary function of stress tensor σ, with a constraint imposed on the volume of the structure $V \equiv |\Omega| \le V_{\max}$,

– the objective is to minimize the volume of the structure

$$J = \int_{\Omega} d\Omega \tag{28}$$

with constraints imposed on equivalent stresses σ_{eq} of the structure

$$\sigma_{eq}(\boldsymbol{x}) \le \sigma_{\max},\ \boldsymbol{x} \in \Omega \tag{29}$$

In order to solve the formulated problem FE models of the structures are considered (Zienkiewicz, 2000). The structure is divided into finite elements $\Omega_e, e = 1, 2, ..., R$, and node displacements are calculated by solving a system of linear algebraic equation

$$\mathbf{KU=F} \tag{30}$$

where **U** is a column matrix of unknown displacements, **F** is a known column matrix of acting forces and **K** is a known global stiffness matrix of the structure whose elements are given as follows:

$$\mathbf{K}^{\mathbf{e}} = \int_{\mathbf{V}^{\mathbf{e}}} \mathbf{B}^{\mathbf{T}}\mathbf{D}\mathbf{B}\mathrm{d}\mathbf{V} \tag{31}$$

where **D** and **B** are the known elasticity and geometrical matrices, respectively, $\mathbf{V}^{\mathbf{e}}$ represents the volume of the finite element.

The distribution of Young's modulus $E_t(\boldsymbol{x}), \boldsymbol{x} \in \Omega_t$ or thickness $g_t(\boldsymbol{x}), \boldsymbol{x} \in \Omega_t$ in the structure is describing by a surface

$$W_E(\boldsymbol{x}), \boldsymbol{x} \in H^2 \text{ or } W_g(\boldsymbol{x}), \boldsymbol{x} \in H^2, \boldsymbol{x} = (x, y) \tag{32}$$

for plate in plane stress/strain, bending plate or a hypersurface

$$W_E(\boldsymbol{x}), \boldsymbol{x} \in H^3 \text{ or } W_g(\boldsymbol{x}), \boldsymbol{x} \in H^3, \boldsymbol{x} = (x, y, z) \tag{33}$$

for shell (Figure 41). The surfaces (hypersurfaces) $W_E(\boldsymbol{x})$ i $W_g(\boldsymbol{x})$ are stretched under $H^d \subset E^d$, $(d = 2,\ 3)$ and the domain Ω_t is included in H^d, i.e. $(\Omega_t \subseteq H^d)$. The shapes of the surfaces (hypersurfaces) $W_E(\boldsymbol{x})$ i $W_g(\boldsymbol{x})$ is controlled by genes hi, i=1,…,N, which create a chromosome

$$ch = [h_1, h_2, ..., h_i, ..., h_N] \ , \quad h^{\min} \le h_i \le h^{\max} \tag{34}$$

where

$h^{\min}$ - the minimum value of the gene,

$h^{\max}$ - the maximum value of the gene.

Genes are values of the function $W_\alpha(\boldsymbol{x}),\ \alpha = E, g$ in interpolation nodes xi, i.e. $h_i = W(\boldsymbol{x}_i)$, i=1,2,…,N.

The assignation of Young's moduli or thickness to each finite element $\Omega_e, e = 1, 2, ..., R$ is adequately performed by the mapping:

$$E_e = W_E(\boldsymbol{x_e}), \boldsymbol{x_e} \in \Omega_e, e = 1, 2, ..., R \tag{35}$$

$$g_e = W_g(\boldsymbol{x_e}), \boldsymbol{x_e} \in \Omega_e, e = 1, 2, ..., R \tag{36}$$

It means that each finite element can have different material. When the value of Young's modulus or thickness for the e-th finite element is included in:

- the interval $0 \le E_e < E_{\min}$ (or $0 \le g_e < g_{\min}$), the finite element is eliminated and the void is created,

- the interval $E_{\min} \le E_e < E_{\max}$ (or $g_{\min} \le g_e < g_{\max}$), the finite element remains having the value of the Young's modulus from this material (Figure 42).

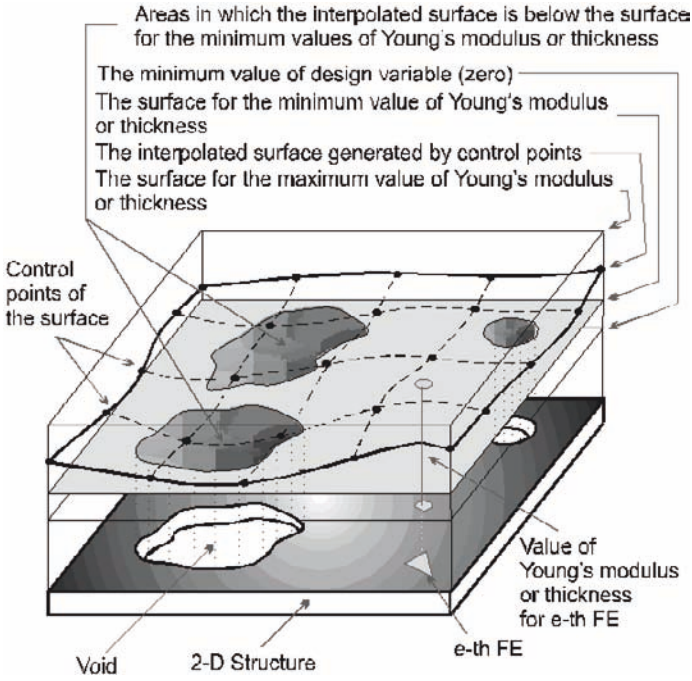


Figure 41. Illustration of the idea of genetic generation for 2-D structure

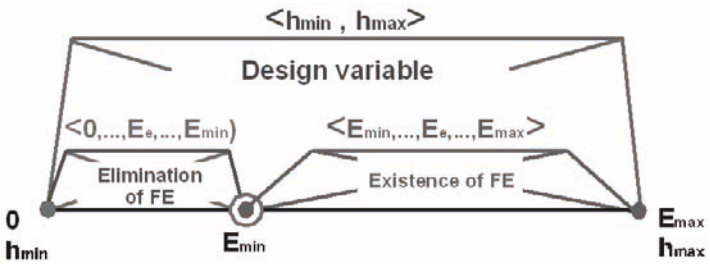


Figure 42. Requirements for elimination and existence of the finite elements

7.3 Interpolation procedures

In the considered work three different interpolation procedures have been applied: (i) interpolation procedure of the function $f(x,y)$ used for solving tasks of the optimization of plates in plane stress/strain and bending plates, (ii) interpolation procedure of the function $f(x,y,z)$ used in the case of the optimization of shells and (iii) two interpolations procedures for 3-D problems.

For optimization problems of the plates in plane stress/strain and bending plates the interpolation surface which is described by the following expression has been applied

$$W_{\alpha}(x,y) = \mathbf{\Phi}\ (\mathbf{X}^{-1} \otimes \mathbf{Y}^{-1}) \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{16} \end{bmatrix} \tag{37}$$

where

$$\mathbf{\Phi} = [1, x, x^2, x^3] \otimes [1, y, y^2, y^3] = [1, y, y^2, y^3, x, xy, xy^2, xy^3, x^2, x^2 y, x^2 y^2, x^2 y^3, x^3, x^3 y, x^3 y^2, x^3 y^3] \tag{38}$$

and X and Y matrices are given as follows

$$\mathbf{X} = \mathbf{Y} = \begin{matrix} 1 & x & x^2 & x^3 \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} \tag{39}$$

In the case of shells optimization problems, the application of the interpolation procedure of the function $f(x,y,z)$, creates some difficulties, which are connected with the curved shape of the shell. There are some difficulties with the adaptation of the procedure to the number of the control points and to the distribution of the control points in the three-dimensional space. In connection with the mentioned difficulties, for the problems of the shells optimization, the interpolation procedure based on the finite element mesh, is introduced (Figure 43). That procedure (Table 12) is grounded on the analysis of the neighbourhoods of the individual nodes.

Table 12. Interpolation procedure

```
Read nodes i=1,2,...,G and elements e=1,2,...,E
For i=1,2,...,G read initial vector p_1^0, p_2^0,..., p_w^0
of the optimization parameters

For k=0,1,2,...,K      * k – iteration step *
{
      For i=1,2,...,G     * for all the nodes *
      {
       p_i^{k+1} = p_i^k
      For j=1,2,...,M      * for all neighbour-nodes of the node i *
      compute max(p_j)
      compute min(p_j)
      compute p_i=1/2[max(p_j)+ min(p_j)]
      }
      For c=1,2,...,C     * for all control points *
      {
       p_c=h_i     * rewriting of the changed parameter values in the
                     control points to the initial values
                     (values of the genes) *
      }
}
```

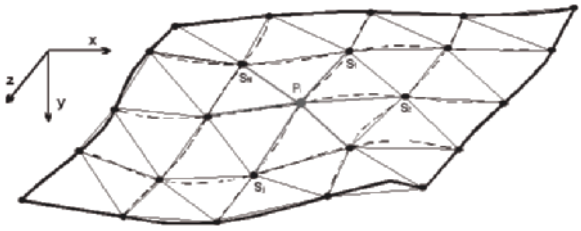


Figure 43. Nodes S_j being in neighbourhood of the node P_i

In the case of 3-D structures the distribution of Young's modulus $E(x,y,z)$, $(x,y,z) \in \Omega_t$ in the structure is described by a hyper surface $W(x,y,z)$, $(x,y,z) \in H^3$. The hyper surface $W(x,y,z)$ is stretched under $H^3 \subset E^3$ and the domain Ω_t is included in H^3, i.e. $(\Omega_t \subseteq H^3)$.

The shape of the hyper surface $W(x,y,z)$ is controlled by genes d_j, j=1,2,…,N, which create a chromosome

$$ch = \langle d_1, d_2, ..., d_j, ..., d_N \rangle \tag{40}$$

Gene values are described by the function $W(x, y, z)$ in interpolation nodes (control points) $(x, y, z)_j$, i.e. $d_j = W\left[(x, y, z)_j\right]$, j=1,2,…,N.

The following constraints are imposed on the genes

$$d_j^{\min} \le d_j \le d_j^{\max} \tag{41}$$

where $d_j^{\min}$ - the minimum value of the gene and $d_j^{\max}$ - the maximum value of the gene.

In the first interpolation – the multinomial interpolation of the hyper surface is expressed as follows

$$W(x, y, z) = \Phi\left(\boldsymbol{X}^{-1} \otimes \boldsymbol{Y}^{-1} \otimes \boldsymbol{Z}^{-1}\right)\begin{bmatrix} d_1 \\ \div \\ d_{27} \end{bmatrix} \tag{42}$$

where

$$\begin{aligned} \Phi = &[1, x, x^2] \otimes [1, y, y^2] \otimes [1, z, z^2] = [1, z, z^2, y, yz, yz^2, y^2, \\ &y^2 z, y^2 z^2, x, xz, xz^2, xy, xyz, xyz^2, xy^2, xy^2 z, xy^2 z^2, \\ &x^2, x^2 z, x^2 z^2, x^2 y, x^2 yz, x^2 yz^2, x^2 y^2, x^2 y^2 z, x^2 y^2 z^2] \end{aligned} \tag{43}$$

and X, Y and Z are matrices described as follows

$$\boldsymbol{X} = \boldsymbol{Y} = \boldsymbol{Z} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \tag{44}$$

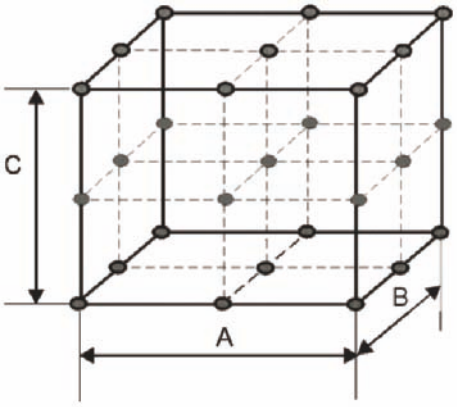


Figure 44. Spacing of control points

The structure which is under the optimization process is inserted into a cube H^3 whose edges have length A=2, B=2, C=2 (Figure 44). The model of the structure is scaled by means of following expressions:

x=(A* coordinate x)/length of the model
y=(B* coordinate y)/width of the model (45)
z=(C* coordinate z)/height of the model

In the second interpolation (Table 13) which works in iterative case – the interpolation bases on the neighborhood of elements - the hyper surface is expressed as follows

$$\mathbf{W^{k+1}} = f(\mathbf{W^k}),\ k = 0,1,2,...,K \tag{46}$$

As an initial approximation, in step k=0 vector is accepted

$$\mathbf{W^0} = [p_1^0, p_2^0, ..., p_i^0, ..., p_E^0],\ i = 1,2,...,R \tag{47}$$

where: $p_i^0,\ i = 1,2,...,R$ are the initial values of the parameter of the optimization for particular elements of mesh, E is a total number of elements. The values of the parameter of the optimization in control points are the values of the genes of chromosome (40).

Next approximations of the vector of parameter optimization

$$\mathbf{W^{k+1}} = [p_1^{k+1}, p_2^{k+1}, ..., p_W^{k+1}],\ k = 0,1,2,...,K \tag{48}$$

Table 13. Interpolation procedure in optimization of 3-D structures

```
Load nodes i=1,2,...,W  and elements e=1,2,...,E
For e=1,2,...,E  load the initial vector of parameters optimization
For k=0,1,2,...,K      „k – step of iteration”
{
For i=1,2,...,E    „for all elements”
{
If T[i]=0
{
For  j=1,2,...,M    „for all neighbouring elements for i – element”
Calculate max(p_j)
Calculate min(p_j)
Calculate p_i^{k+1}=1/2[max(p_j^k)+ min(p_j^k)]
}
If T[i]=1  p_i^{k+1}= p_i^k
}
}
```

are calculated by equation

$$p_i^{k+1} = \frac{1}{2}[\max(p_j^k) + \min(p_j^k)],\ j = 1,2,...,M \tag{49}$$

where:

T[i] – vector determining position of control points (if T(i) is equal one –*i-th* element contains control point, if T(i) is equal zero – *i-th* element does not contain control point).

M – number of neighbours $S_j,\ j = 1,2,...,M$ for i-th element $P_i,\ i = 1,2,...,R$ (Figure 45),

p_i^{k+1} – value of the parameter of the optimization for *i-th* element, in step k+1,

p_j^k – value of the parameter of the optimization for j-th element which is neighbour for element *i-th*, in step *k-th*,

$\max(p_j^k)$ – maximal value of the parameter of the optimization for elements which are neighbours for element *i-th*, in step *k-th*,

$\min(p_j^k)$ - minimal value of the parameter of the optimization for elements which are neighbours for element *i-th*, in step *k-th.*

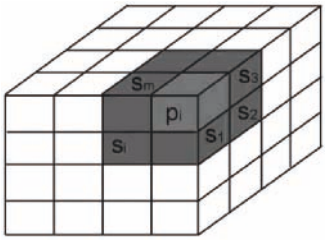


Figure 45. Elements S_j neighbour with element P_i

For the first interpolation a part of working space may not be used. The second interpolation has not this disadvantage. Moreover for the first interpolation the number of control points is constant, for the second interpolation, the optional number of control points can be loaded (Figure 46).

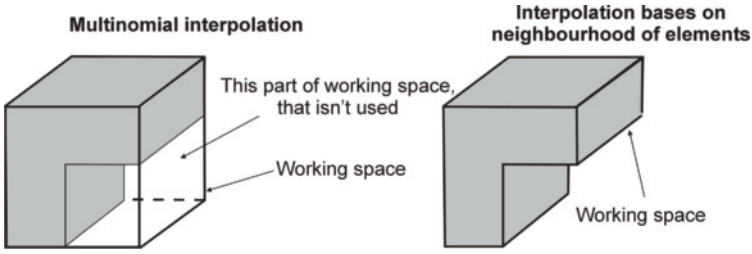


Figure 46. Working space for two different interpolations

Minimization of the fitness functions with respect to the chromosome (40) is performed by means of an evolutionary algorithm with the floating point representation.

After the FEM discretization the starting population of chromosomes is randomly generated. At the next step the main loop of the optimization algorithm is performed.

Operations included in the main loop lead to the calculation of the fitness function. It requires that the boundary-value problem should be solved by the FEM. After the calculation of the fitness function for all chromosomes in the population the evolutionary algorithm is applied. The evolutionary algorithm contains the following operators: the ranking selection, the simple and arithmetical crossovers, the uniform and boundary mutations and the cloning (Michalewicz, 1996). As the result a new offspring population is created.

In the optimization algorithm the method of the penalty function is applied. The penalty function is taken in the form of "penalty of die". In the consequence unacceptable chromosomes are eliminated, namely chromosomes which do not fulfil the introduced constraints.

The end of the algorithm's work, i.e. a break in the main loop activity, occurs after the declared generation number. The algorithm can be also stopped, when after the specified number of iteration the change of the fitness function is very small.

The assignation of Young's moduli to each finite element Ω_e, $e = 1,2,...,R$ is performed by the mapping:

$$E_e = W\left[(x,y,z)_e\right], (x,y,z)_e \in \Omega_e, e = 1,2,...,R \tag{50}$$

It means that each finite element can have different material. The procedure of eliminating finite elements and creating new voids accordingly to values of E_e is the same as described before (Figure 42).

7.4 Additional procedure aiding of evolutionary optimization

In order to improve the optimization process, an additional procedure is introduced (Figure 47). Values σ_{min} and p (adequately minimum stress and stress increment) are input data to the procedure. The accuracy of obtained solutions depends on the ascribed values of σ_{min} and p. Small values of σ_{min} and p guarantee the more precise solution but it is compensated by the long computation time.

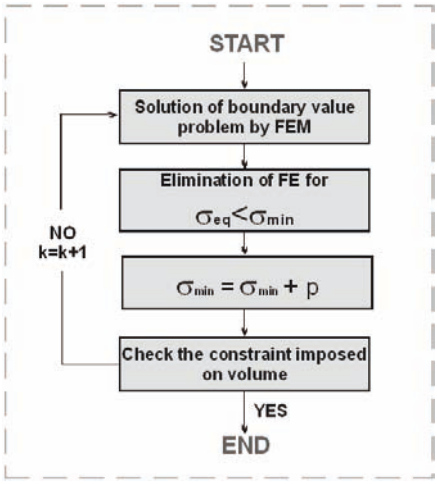


Figure 47. Additional procedure aiding of evolutionary optimization (σ_{min} - minimum stress, p – stress increment)

Implementation of this procedure increases:

- the number of chromosomes that fulfil imposed constraints,
- the effectiveness of the optimization algorithm by removing unnecessary material which is not strained enough.

Besides additional procedure facilitates the smooth shape of the structure boundary.

7.5 Assignation of materials

The optimization process based on controlling Young's moduli allows to find the optimal solution in which each finite element can have a different value of the Young's modulus. However, in practice structures are made from a specified number of materials. Therefore the range of the Young's modulus value should be divided into the enforced number of subintervals (equal to the number of materials). Each subinterval represents another material whose the Young's modulus belongs to this subinterval. The subintervals should have the same length and their centres correspond with Young's modulus values of suitable materials. The example of this idea for prescribed 3 different materials is illustrated in Figure 48.

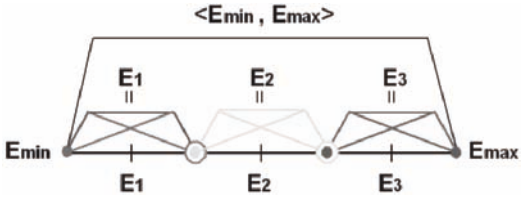


Figure 48. Idea of introduction of three materials

7.6 Evolutionary optimization algorithm of topology, shape and material or thickness

Minimization of the fitness functions Eq. (27) or (28) with respect to the chromosome (34) is performed by means of an evolutionary algorithm with the floating point representation (Figure 49). After the FEM discretization the starting population of chromosomes is randomly generated. At the next step the main loop of the optimization algorithm is performed. Operations included in the main loop lead to the calculation of the fitness function. It requires that the boundary-value problem should be solved by FEM. After the calculation of the fitness function for all of the chromosomes in the population the evolutionary algorithm is applied. As the result a new offspring population is created. The end of the algorithm's work, i.e. a break in the main loop activity, occurs after the

declared generation number. The algorithm can be also stopped, when through the specified iteration number the change of the fitness function is very small.

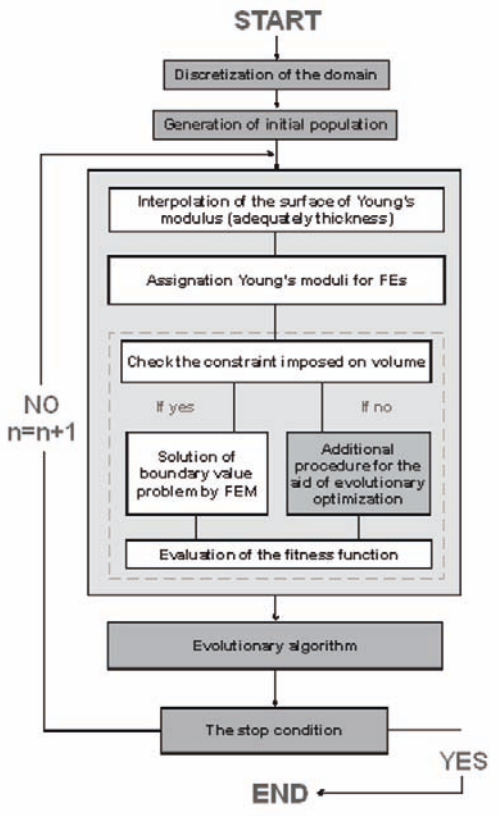


Figure 49. Evolutionary optimization of 2-D and 3-Dstructures

7.7 Examples of topological evolutionary optimization of 2D structures

Three numerical examples are considered for 2-D problems. The structures are discretized by triangular finite elements and subjected to the volume or stress constraints. The results of the examples are obtained by use of described optimization method based on sequential evolutionary algorithm of parameters included in Table 14. To solve the boundary value problem a professional program of finite element method – MSC NASTRAN is used. Using proposed method, material properties or thickness of finite elements are changing evolutionally and some of them are eliminated. As a result the optimal shape, topology and material or thickness of the structures are obtained.

Table 14. Parameters of sequential evolutionary algorithm

Number of chromosomes	100
Probability of cloning	2%
Probability of uniform mutation	5%
Probability of boundary mutation	5%
Probability of simple crossover	10%
Probability of arithmetical crossover	10%
Selection method	rang selection

Example 9 – optimization of the bicycle frame. The task of optimization of shape, topology and distribution of three different materials of the bicycle frame by minimization of the stress functional and with the volume constraint is considered. The size of the initial system and boundary conditions are prescribed according to the person of the mass of 90 kg (Figure 50a).

Table 15. Input data to the optimization task (*Example 9*)

dimensions a; b; c; d [mm]	forces [kN]	number of design variables	range of E_e [10^5 MPa]; existence or elimination of the finite element
900 ; 610; 440; 300	Q_1=1.0, Q_2=0.15, Q_3=5.0, Q_4=7.5, Q_5=0.1, Q_6=0.75	16	$0.5 \le E_e < 0.75$ elimination $0.75 \le E_e \le 2.25$ existence

thickness of the plate [mm]	σ_{min} ; p [MPa]	materials [10^5 MPa]	V_{max} [cm^3]
2.0	3.0 ; 3.0	Material 1. E=1.0 for $0.75 \le E_e < 1.25$ Material 2. E=1.5 for $1.25 \le E_e < 1.75$ Material 3. E=2.0 for $1.75 \le E_e \le 2.25$	450

In the present task the set of 16th control points of the interpolation surface has been introduced (Figure 50b). Input data to the optimization program and parameters of the evolutionary algorithm are presented in Table 15.

a) b)

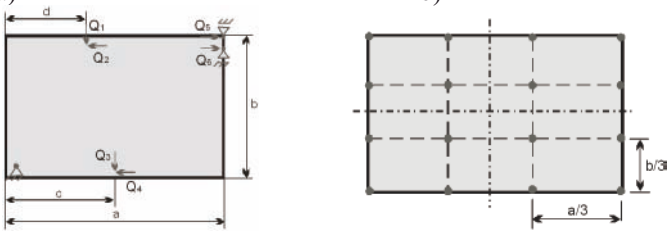


Figure 50. A bicycle frame: a) the initial geometry with scheme of loading; b) distribution of the control points of the interpolation surface

The results of the optimization are presented as the map of materials distribution (Figure 51a), the map of stresses (Figure 51b), the shape of interpolation surface (Figure 51c) and the model of the bicycle (Figure 51d) for the best obtained solution.

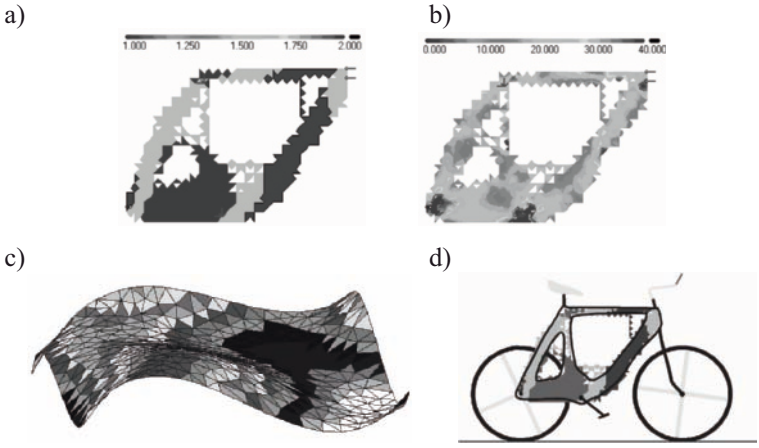


Figure 51. The results of the evolutionary optimization of a bicycle frame. The best individual for the t=50th generation: a) distribution of three different materials; b) the map of stress; c) the shape of interpolation surface; d) model of the bicycle based on optimal frame

Example 10 – optimization of the bending plate. The task of the optimization of shape, topology and thickness of the bending plate by minimization of the volume functional and with the stress constraint is considered. A plate is loaded with the concentrated force $Q_1 \div Q_5$ and fixed on the boundary (Figure 52a). In the present task the set of 16th control points of the interpolation surface has been introduced (Figure 52b). Input data to the optimization program and parameters of the evolutionary algorithm are included in Table 16. The results of the optimization are presented as the map of thickness (Figure 53a), the map of stresses (Figure 53b) and the shape of interpolation surface (Figure 53c) for the best obtained solution.

Table 16. Input data to the optimization task (*Example 10*)

dimensions a; b [mm]	forces [kN]	number of design variables	material [10^5 MPa]	
600; 200	$Q_1=Q_2=Q_3=Q_4=0.6$, $Q_5=1.2$	16	E=2.0	
range of g_e [mm]; existence or elimination of the finite element			σ_{min} [MPa]	σ_{max} [MPa]
$3.0 \le g_e < 5.0$ elimination; $5.0 \le g_e \le 15.0$ existence			4.0	100.0

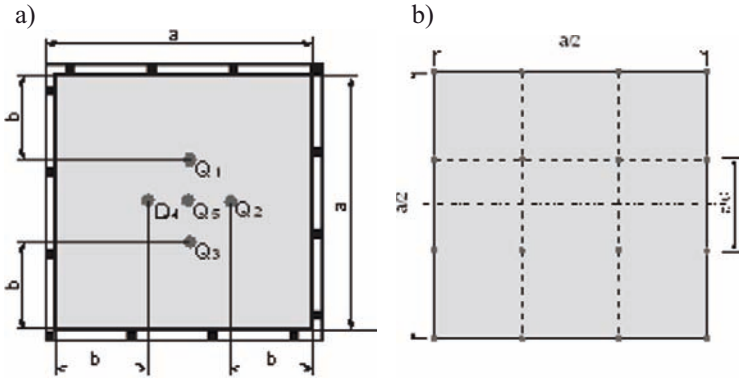


Figure 52. A bending plate:
a) the initial geometry with scheme of loading; b) distribution of the control points of the interpolation surface (¼ part of the geometry)

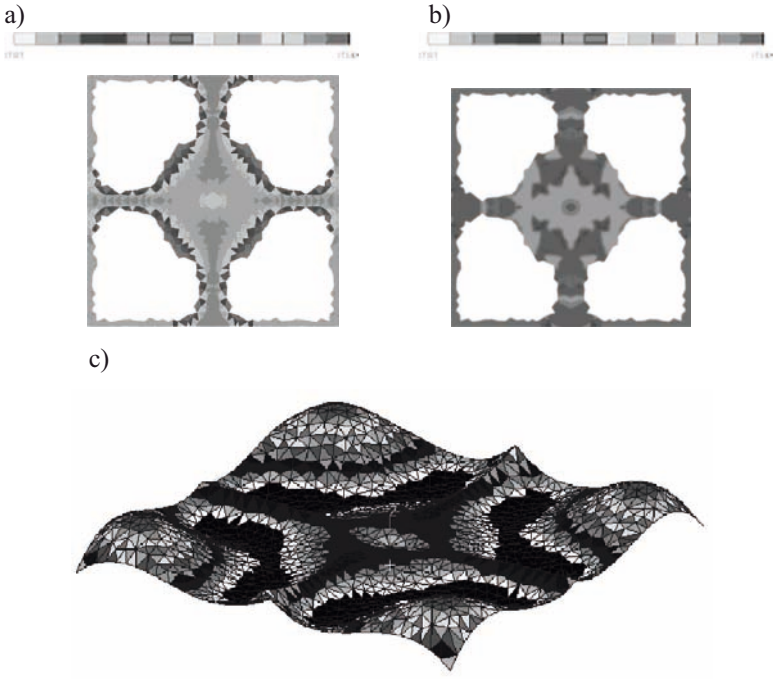


Figure 53. The results of the evolutionary optimization of the bending plate. The best individual for the t=50th generation. a) the map of thickness; b) the map of stress; c) the shape of interpolation surface

Example 11 – Optimization of a car wheel. The task of the optimization of shape, topology and thickness of a car wheel by the minimization of the stress functional and with the volume constraint is considered.

A car wheel geometry of characteristic dimensions, included in Table 17, is built from three surfaces of revolution (Figure 54): the central surface with the holes destined for the fastening bolts, the surface of the ring of the wheel and the surface connecting the two mentioned earlier. The last one is subjected to the optimization process. The shell-structure is loaded with the tangent force s_0 (torsion of the wheel) and with a pressure c_0 (pressure in the tyre).

Table 17. Characteristic dimensions of a car wheel

diameter of the wheel LW	355.6 mm
width of a tyre LF	175 mm
diameter of the wheels spacing LK	110 mm
diameter of a wheel hub LP	60 mm
thickness of the wheel hub	30 mm
thickness of a tyre	8 mm

The loadings are applied to the ring of the wheel (Figure 55a). The structure is stiffly supported around the holes destined for the fastening bolts and is also supported on the central surface in the direction of the rotation axis of the wheel (Figure 55a). In the considered task the symmetry of the car wheel (revolution of the 1/5 part of the structure) during the distribution of the control points of the interpolation hypersurface has been used (Figure 55b). In this way the number of design variables (genes) could be decreased and the symmetrical results could be obtained. This reasoning is purposeful because of the necessity of the car wheel balance.

Input data to the optimization task and the parameters of evolutionary algorithm are included in the Tables 18. The results of the optimization are presented as the maps of thickness (Figure 57a) and the maps of stresses (Figure 57b) for the best obtained solutions. Figure 56 shows the evolution of the best individual in chosen generations on the example of the evolutionary optimization of a car wheel.

Table 18. Input data to the optimization task (*Example 11*)

tangent force s^0 [N]	pressure c^0 [Mpa]	number of design variables	number of control points	V_{max} [cm^3]
500	0.22	23	86	5 500
material	Range of g_e [mm]	existing of an element	elimination of an element	
Aluminium	$4 \le g_e \le 20$	$4 \le g_e < 10$	$10 \le g_e \le 20$	

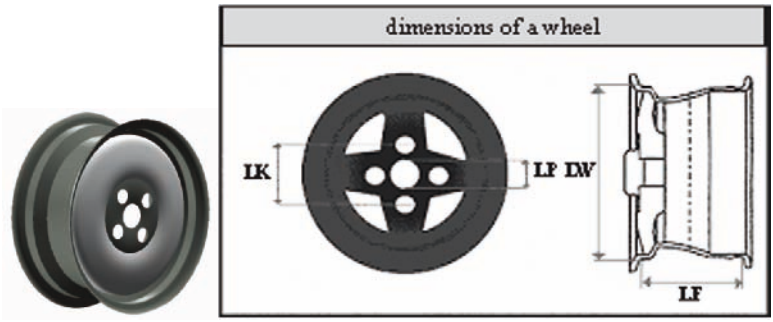


Figure 54. Geometry and characteristic dimensions of a car wheel

a) b)

Figure 55. A car wheel: a) the kind of the loading and supporting of the shell; b) distribution of the control points of the interpolation hypersurface

7.8 Numerical example for 3-D structures

Example 12. A 3-D „L" structure (Figure 58a) with dimensions and loadings given in Table 19 is optimized for the criterion of minimum of the volume with constraints imposed on stresses and displacements (Table 20). The Table 21 contains input data. The optimize structures is discretized by cubic finite elements. The parameters of evolutionary algorithm are included in Table 22 and material data in Table 23.

Table 19. The dimensions and loading of 3-D structure

Dimensions [mm]	
A	48
B	48
C	24
D	24
E	24
Loading [kN]	
Q	8.45

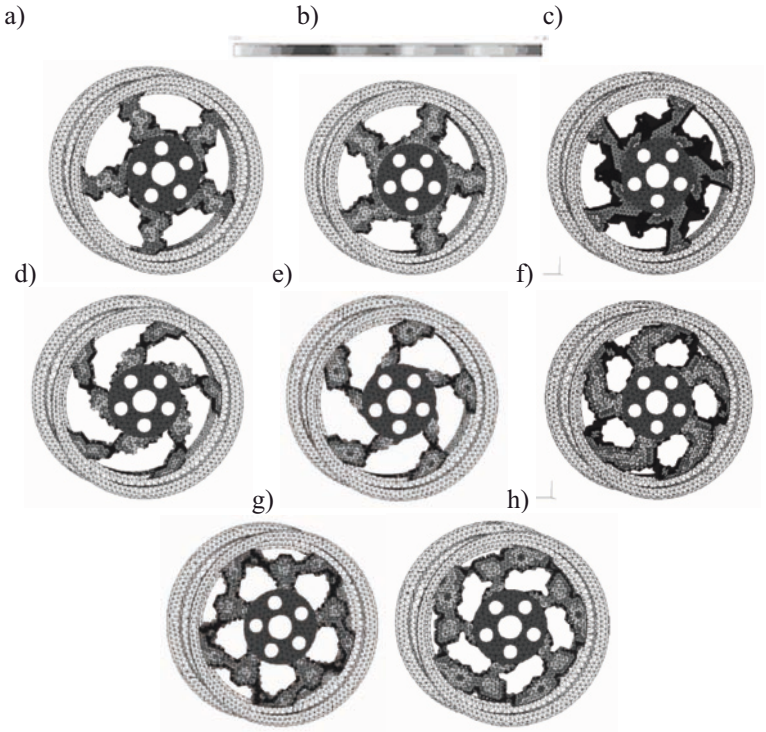


Figure 56. Evolution of the best individual in chosen generations on the example of the evolutionary optimization of a car wheel. The best individual in the generation No.: a) 1, b) 3, c) 6, d) 9, e) 12, f) 17, g) 45, h) 100

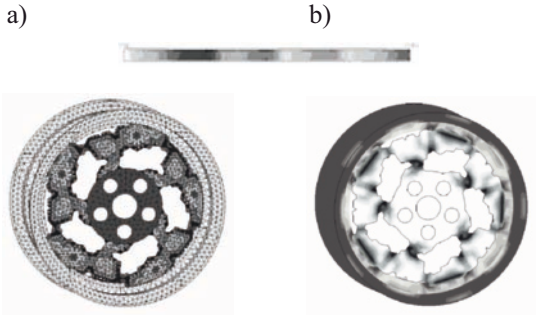


Figure 57. The results of the evolutionary optimization of a car wheel. The best individual for the t=100th generation: a) the map of thickness; b) the map of stresses

Computational results obtained after 73 generations are presented in the form of a map of the distribution of Young's moduli (Figure 58 b, c), stresses (Figure 58 d, e) and a map of displacements (Figure 58 f, g). The structure after smoothing process is presented in Figure 59.

Table 20. Constraints

Constraints		
Dimensions of cubicoid	Maximal stress	Genes 1÷27
48 x 48 x 24	600 MPa	0 ÷ 1
	Maximal displacement	
	0.08 mm	

Table 21. Input data

Minimal Young's module	Numbers of chromosomes
0.4 x $2*10^5$ MPa	80
Type of interpolation	Step of iteration in smooth procedure
multinomial interpolation	25

Table 22. The parameters of evolutionary algorithm

Parameters of evolutionary algorithm		
Numbers of design variables	Numbers of generations	
27 – multinomial interpolation 11 - interpolation bases on the neighborhood of elements	2000	
Probability of cloning	Probability of uniform mutation	Probability of boundary mutation
10 %	5 %	2 %
Probability of simple crossover	Probability of arithmetical crossover	
10 %	10 %	

Table 23. Material data

Material data	
Poisson ratio	Young's module (E_{max})
0.3	$2e5$ MPa

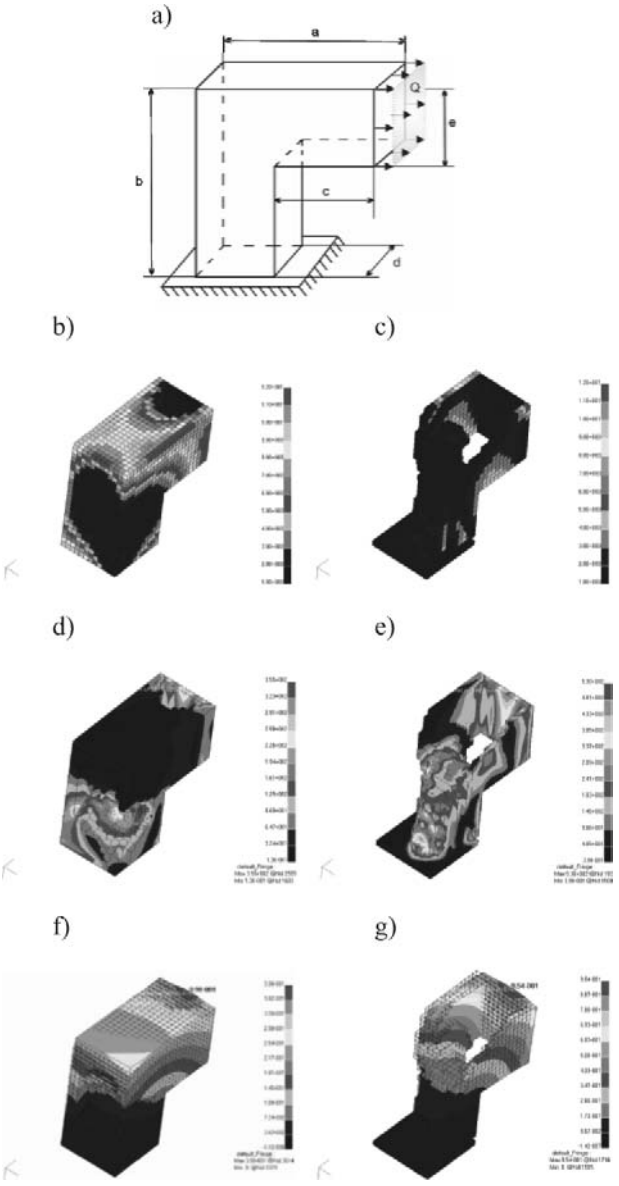


Figure 58. L structure: a) the scheme of loading, b), c) distribution of Young's moduli, d), e) map of stresses f) g) map of displacements b), d), f) the best solution after first generation, c), e), g) the best solution after optimization

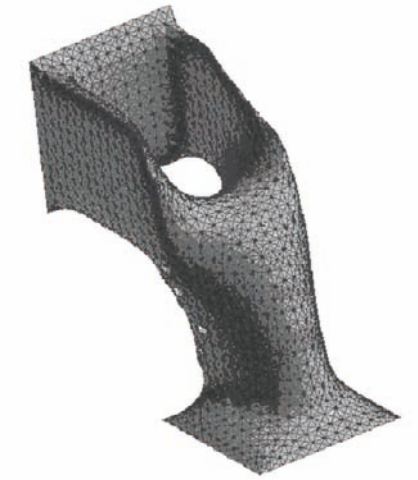


Figure 59. The optimal L structure after smoothing

8 Evolutionary Multiobjective Optimization

8.1 Introduction

In many real-world engineering problems several aims must be satisfied simultaneously in order to obtain an optimal solution. In the first phase of the design process the set of objectives is unclear and the designer has to define them as precisely as possible. Moreover, for the multiobjective optimization (Augusto *et al*, 2006), (Coello, 1999, 2000), (Deb, 1999) the goals are usually in conflict with each other. For example, the volume of the radiator should be minimized while the total dissipated heat flux or maximal value of the equivalent stress should be maximized (or minimized also). The common approach in this sort of problems is to choose one objective (for example the volume of the structure) and incorporate the other objectives as constrains. This approach has been presented in previous works Białecki *et al* (2005), Burczyński and Długosz (2002, 2006) and Długosz (2001), but it has the disadvantage of limiting the choices available to the designer, making the optimization process rather difficult.

The evolutionary algorithms using the Pareto approach are proposed as the optimization technique. The fitness function is calculated for each chromosome in each generation by solving a boundary value problem of thermoelasticity by means of the FEM (Beer, 1983). The optimized radiators are modelled as structures subjected to mechanical and thermal boundary conditions. The interaction of stress and temperature fields is modelled by means of the theory of the thermoelasticity.

8.2 Multiobjective optimization

In the multiobjective optimization solution of the problem is represented more than one objective function. In such problems all of the objective functions cannot be simultaneously improved, moreover they are usually in conflict with each other, so the term "optimize" means finding such a solution which would give the values of all objective functions acceptable to the designer. Instead of one optimal solution the set of optimal solutions can be received. A multiobjective optimization problem can be expressed as follows:

find the vector $\mathbf{x} = [x_1, x_2, \ldots x_n]^T$ which will satisfy the m inequality constrains:

$$g_i(\mathbf{x}) \geq 0 \qquad i = 1, 2, \ldots, m \tag{51}$$

and the p equality constrains

$$h_i(\mathbf{x}) \geq 0 \qquad i = 1, 2, \ldots, p \tag{52}$$

which minimizes the vector of k objective functions

$$f(\mathbf{x}) = \left[f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}) \right]^T \tag{53}$$

The set of objective functionals for multiobjective optimization of radiators can be defined as:

– minimization the volume of the structure:

$$\min_{\mathbf{x}} V(\mathbf{x}) \tag{54}$$

– minimization of the maximal value of the equivalent stress:

$$\min_{\mathbf{x}} \sigma_{eq}^{\max}(\mathbf{x}) \tag{55}$$

– maximization of the total dissipated heat flux

$$\max_{\mathbf{x}} q(\mathbf{x}) \tag{56}$$

In the present work taking advantage of the Pareto concept, the multiobjective optimization is performed. The Pareto-optimality is defined as a set $\mathcal{F}_P$, where every element f_P is a solution of the problem defined by (51)-(53), for which no other solutions can be better with regard to all objective functions. In other words the solution is Pareto optimal if there exist no feasible vector

decrease some criterion without causing a simultaneous increase of another criterion. In Figure 60 a bold line is used to marked the set of Pareto optimal solutions which is called the *Pareto front*.

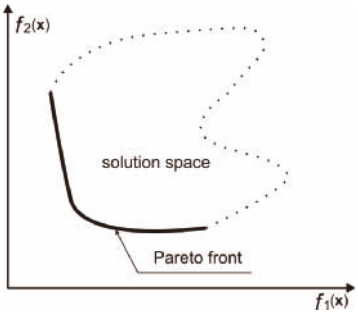


Figure 60. An example of the biobjective problem

Considering two solutions vector $\mathbf{x}$ and $\mathbf{y}$ for a minimization problem, $\mathbf{x}$ is contained in the Pareto front if:

$$\begin{aligned} &\forall i \in 1, 2, \ldots, k: \ f_i(\mathbf{x}) \le f_i(\mathbf{y}) \\ &\textit{and} \\ &\exists j \in 1, 2, \ldots, k: \ f_j(\mathbf{x}) < f_i(\mathbf{y}) \end{aligned} \tag{57}$$

The Pareto optimum always gives not a single solutions, but a set of solutions called non-dominated solutions or efficient solutions.

8.3 Multiobjective evolutionary algorithm

In order to solve the optimization problem the evolutionary algorithm (Arabas, 2001), (Michalewicz, 1996) with the real-coded representation has been proposed. The solution of this problem is given by the best chromosome whose genes represent design parameters responsible for shape of heat radiator. The flow chart of the multiobjective evolutionary algorithm is shown in Figure 61.

The proposed evolutionary algorithm starts with a population of chromosomes randomly generated. Two kinds of the mutation are applied: the uniform mutation and the Gaussian mutation. The operator of the uniform mutation replaces a randomly chosen gene of the chromosome with the new random value. This value corresponds to the design parameter with its constrains. For the Gaussian mutation a new value of the gene is created with the use of Gaussian distribution. The probability of the mutation decides how many genes will be modified in each population The operator of the simple crossover

creates two new chromosomes from the two randomly selected chromosomes. Both chromosomes are cut in randomly position and merge together. In order to compute k objective functions the thermoelasticity problem is solved.

The selection is performed on the base of a ranking method, information about Pareto optimal solutions and the similarity of solutions. This procedure is very similar to the method of selection proposed by Fonseca and Fleming (1995).

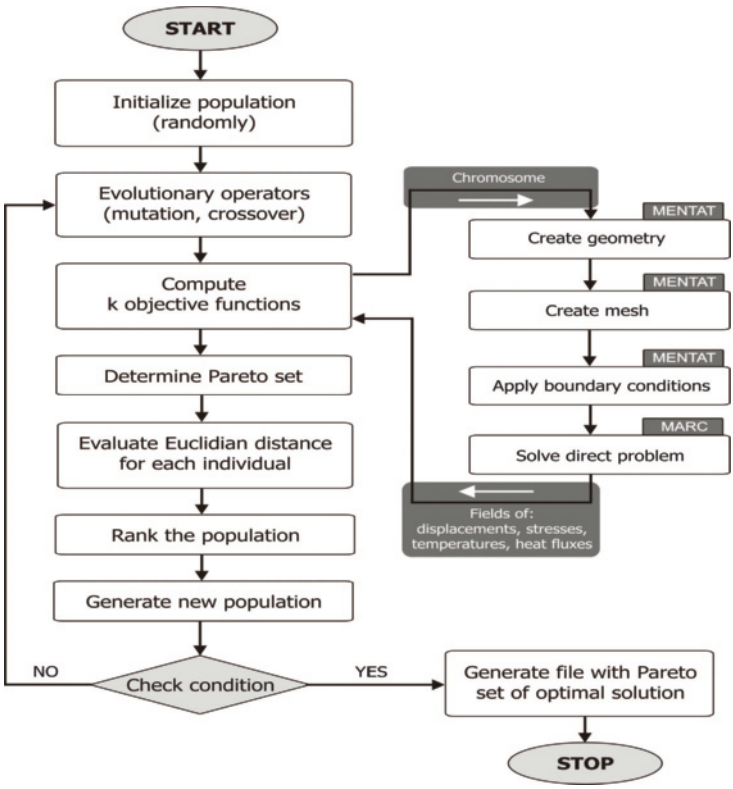


Figure 61. The flow chart of the multiobjective evolutionary algorithm

The Pareto set is determine in the current population using by (57). The Euclidian distance between all chromosomes is defined as follows:

$$\mathrm{ED}(x_i; x_j) = \sqrt{\sum_{n=1}^{popsize} \left(x_i(n) - x_j(n) \right)^2} \tag{58}$$

The rank of each chromosome depends on the number of individuals by which is dominated and scaled value of the Euclidian distance. This scheme helps to conserve diversity in the population. The most similar chromosomes have less probability to survive.

The next iteration is performed if the stop condition is not fulfilled. The stop condition is expressed as the maximum number of iterations. The Pareto set in each generation is stored into file. On the basis of this files the collective Pareto set of optimal solution is generated.

8.4 Evaluation of the fitness function

The fitness function is computed with the use of the steady-state thermoelsticity. Elastic body occupied the domain Ω bounded by the boundary Γ is considered (Figure 62)

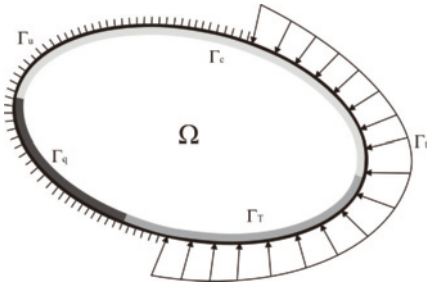


Figure 62. Elastic structure subjected to thermomechanical boundary conditions

The governing equations of the linear elasticity and steady-state heat conduction problem is expressed by the following equations:

$$G\,u_{i,jj} + \frac{G}{1-2v}u_{j,ji} + \frac{2G(1-v)}{1-2v}\alpha T_{,i} = 0 \tag{59}$$

$$kT_{,ii} + Q = 0 \tag{60}$$

where G is a shear modulus and v is a Poisson ratio, u_i is a field of displacements, α is heat conduction coefficient, k is a thermal conductivity, T is a temperature and Q is an internal heat source.

The mechanical and thermal boundary conditions for the equations (59) and (60) take the form:

$$\Gamma_t : t_i = \bar{t}_i \ ; \Gamma_u : u_i = \bar{u}_i$$
$$\Gamma_T : T_i = \bar{T}_i \ ; \Gamma_q : q_i = \bar{q}_i \ ; \Gamma_c : q_i = \alpha(T_i - T^\infty) \tag{61}$$

where $\bar{u}_i, \bar{t}_i, \bar{T}_i, \bar{q}_i, \alpha, T^\infty$ is known displacements, tractions, temperatures, heat fluxes heat conduction coefficient and ambient temperature respectively.
Separate parts of the boundaries must fulfil the following relations:

$$\Gamma = \Gamma_t \cup \Gamma_u = \Gamma_T \cup \Gamma_q \cup \Gamma_c$$
$$\Gamma_t \cap \Gamma_u = \varnothing \tag{62}$$
$$\Gamma_T \cap \Gamma_q \cap \Gamma_c = \varnothing$$

In order to solve numerically thermoelasticity problem finite element method is proposed. After discretization taking into account boundary conditions following system of linear equations can be obtained:

$$\mathbf{KU = F}$$
$$\mathbf{ST = R} \tag{63}$$

where **K** denotes stiffness matrix, **S** denotes conductivity matrix, **U**, **F**, **T**, **R** contain discretized values of the boundary displacements, forces, temperatures and heat fluxes.

This problem is solved by the FEM software – MENTAT/MARC (MSC.MARC 2001). The preprocessor MENTAT enables the production of the geometry, mesh, material properties and settings of the analysis. In order to evaluate the fitness function for each chromosome following four steps must be performed:

Step 1 *(generated using MENTAT)*
Create geometry and mesh on the base of the chromosome genes
Step 2 *(generated using MENTAT)*
Create the boundary conditions, material properties, settings of the analysis
Step 3 *(solved using MARC)*
Solves thermoelasticity problem
Step4
Calculate the fitness functions values on the base of the output MARC file

8.5 Numerical example

Example 13. Consider a radiator whose cross-section is shown in Figure 63a. The structure is made of copper of following material properties: Young modulus E=110000Mpa, Poisson ratio $n = 0.35$, thermal expansion coefficient $a = 16.5 \times 10^{-6}\ 1/K$ and thermal conductivity $k = 380\ W/mK$. Six design variables are assumed: the length of each fin (Z1-Z4), the width of the fins (the same for all fins – Z5) and thickness Z6. The geometry of the radiator is symmetric. The total width of the radiator is equal to 0.1m. Table 24 contains limitations of the design variables. Figure 63b shows thermo-mechanical boundary conditions. Force $P = 10N$ is applied on each fin. The temperature T^0, ambient temperature T^{ot} and the heat convection coefficient α is equal to $100°C, 25°C, 20\ W/mK$, respectively. The multiobjective problem is to determine the specific dimensions of the structure which minimizes the set of proposed functionals (54)-(56).

Table 24. The admissible values of the design parameters.

Design variable	Min value [m]	Max value [m]
Z1, Z2, Z3, Z4	0.01	0.05
Z5	0.0025	0.006
Z6	0.0025	0.008

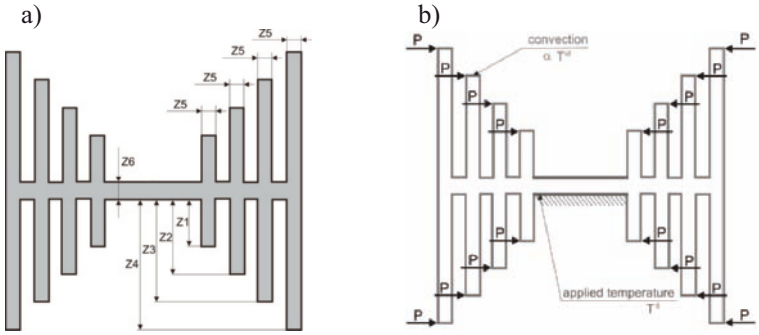


Figure 63. a) The design variables, b) The geometry and the boundary conditions

Several numerical experiments were performed. The set of Pareto optimal solutions with an example of the obtained shape for the minimization both: the volume of the radiator (f_1) and the maximal value of the equivalent stresses (f_2) is presented in Figure 64a. Figure 64b contains the results for the maximization of the total dissipated heat flux and the minimization of the equivalent stresses

(f_2) simultaneously. The set of Pareto solutions obtained for three proposed criterion (f_1 – volume, f_2 – equivalent stress, f_3 – heat flux), are presented in Figure 65.

a) b)

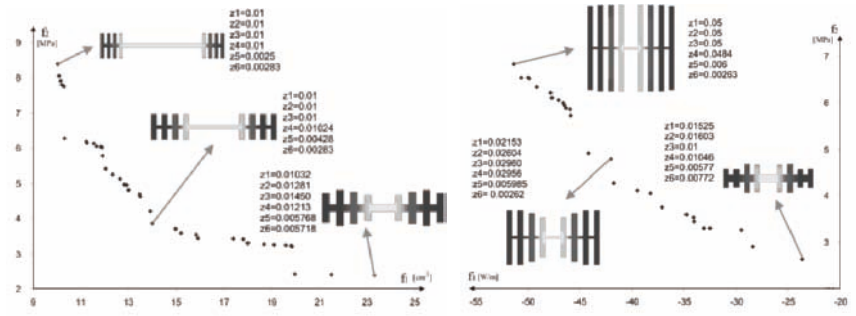


Figure 64. The set of Pareto optimal solution for the two criterion

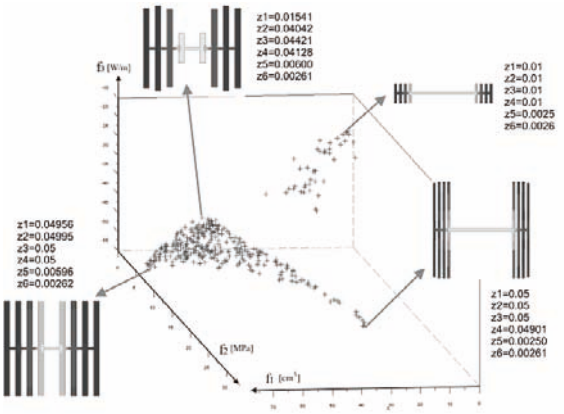


Figure 65. The set of Pareto optimal solution for the three criterion

8.6 Concluding remarks

The multiobjective shape optimization of heat radiators has been presented in the section. The proposed multiobjective evolutionary algorithm gives the designer the set of optimal solutions based on more than one criterion. The choice of one objective and incorporate the other objectives as constrains requires performing optimization many times with different values of the constrains. Such approach makes the optimization process rather inadequate and difficult. Proposed

approach is also more convenient than, for instance, to the "weighting method" in which fitness function is defined as a sum of objective functions and appropriate weights.

9 Immune Optimization

9.1 Introduction

The section deals with an application of global optimization method like the artificial immune system to the optimization problems. The main feature of these methods is to simulate biological processes. The artificial immune system is based on the mechanism discovered in biological immune systems. The main advantage of artificial immune system is the fact that these approach does not need any information about the gradient of the fitness function and gives a strong probability of finding the global optimum. The main drawback of the approach is the long time of calculations.

9.2 Artificial immune system

The artificial immune systems (Castro and Timmis, 2003), (Wierzchoń, 2001) are developed on the basis of a mechanism discovered in biological immune systems. The biological immune system is a complex system which contains distributed groups of specialized cells and organs. The main purpose of the immune system is to recognize and destroy pathogens - funguses, viruses, bacteria and improper functioning cells. The lymphocytes cells play a very important role in the immune system. The lymphocytes are divided into several groups of cells. There are two main groups B and T cells, both contains some subgroups (like B-T dependent or B-T independent). The B cells contain antibodies, which could neutralize pathogens and are also used to recognize pathogens. There is a big diversity between antibodies of the B cells, allowing recognition and neutralization of many different pathogens. The B cells are produced in the bone marrow in long bones. A B cell undergoes a mutation process to achieve big diversity of antibodies. The T cells mature in thymus, only T cells recognizing non self cells are released to the lymphatic and the blood systems. There are also other cells like macrophages with presenting properties, the pathogens are processed by a cell and presented by using MHC (Major Histocompatibility Complex) proteins. The recognition of a pathogen is performed in a few steps (Figure 66).

First, the B cells or macrophages present the pathogen to a T cell using MHC (Figure 66b), the T cell decides if the presented antigen is a pathogen. The T cell gives a chemical signal to B cells to release antibodies. A part of stimulated B cells goes to a lymph node and proliferate (clone) (Figure 66c). A part of the B cells changes into memory cells, the rest of them secrete antibodies

into blood. The secondary response of the immunology system in the presence of known pathogens is faster because of memory cells. The memory cells created during primary response, proliferate and the antibodies are secreted to blood (Figure 66d). The antibodies bind to pathogens and neutralize them. Other cells like macrophages destroy pathogens (Figure 66e). The number of lymphocytes in the organism changes, while the presence of pathogens increases, but after attacks a part of the lymphocytes is removed from the organism.

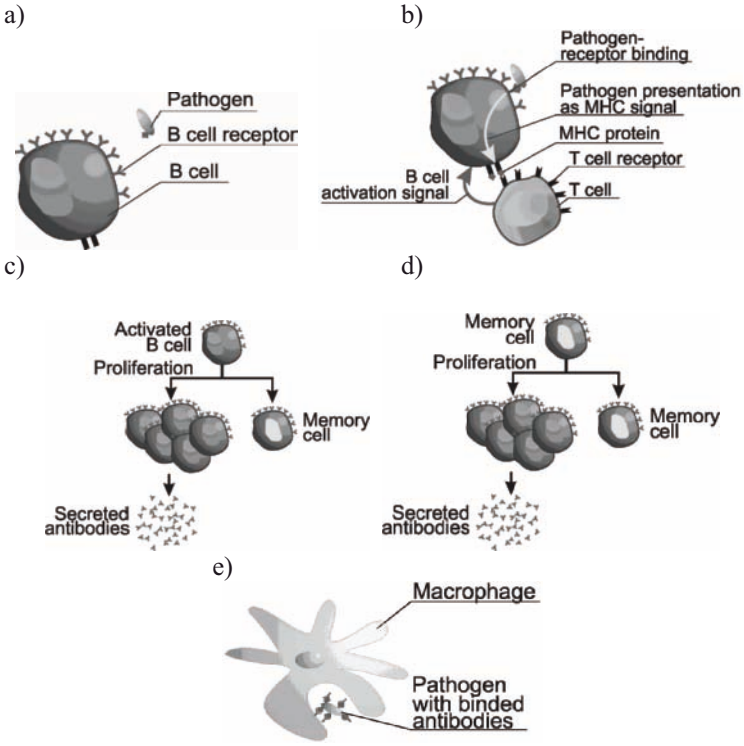


Figure 66. An immune system, a) a B cell and pathogen, b) the recognition of pathogen using B and T cells, c) the proliferation of activated B cells, d) the proliferation of a memory cell – secondary response, e) pathogen absorption by a macrophage

The artificial immune systems (AIS) (Castro and Timmis, 2003) take only a few elements from the biological immune systems. The most frequently used are the mutation of the B cells, proliferation, memory cells, and recognition by using the B and T cells. The artificial immune systems have been used to optimization problems, classification and also computer viruses recognition. The cloning

algorithm Clonalg uses some mechanisms similar to biological immune systems to global optimization problems. The unknown global optimum is the searched pathogen. The memory cells contain design variables and proliferate during the optimization process. The B cells created from memory cells undergo mutation. The B cells evaluate and better ones exchange memory cells. In Wierzchoń (2001) version of Clonalg the crowding mechanism is used - the diverse between memory cells is forced. A new memory cell is randomly created and substitutes the old one, if two memory cells have similar design variables. The crowding mechanism allows finding not only the global optimum but also other local ones. The presented approach is based on the Wierzchoń (2001) algorithm, but the mutation operator is changed. The Gaussian mutation is used instead of the nonuniform mutation in the presented approach.

The Figure 67 presents the flowchart of an artificial immune system.

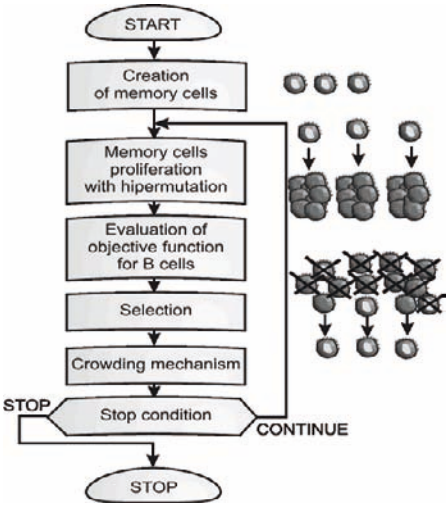


Figure 67. An artificial immune system

The memory cells are created randomly. They proliferate and mutate creating B cells. The number of *nc* clones created by each memory cell is determined by the memory cells objective function value. The objective functions for B cells are evaluated. The selection process exchanges some memory cells for better B cells. The selection is performed on the basis of the geometrical distance between each memory cell and B cells (measured by using design variables). The crowding mechanism removes similar memory cells. The similarity is also determined as the geometrical distance between memory cells. The process is iteratively repeated until the stop condition is fulfilled. The stop condition can be expressed as the maximum number of iterations.

9.3 Comparison of EA and AIS

The numerical examples present the comparison between an artificial immune system and the sequential and distributed evolutionary algorithms. The comparison is performed on the base of the optimization of the known mathematical functions, i.e.: the Branin function with 2 design variables, the Goldstein-Price function with 2 design variables, the Rastrigin function with 20 design variables, and the Griewangk function with 20 design variables (Figure 68), for the best parameters of the algorithms (detected earlier for these functions).

In order to find the optimal parameters of the artificial immune system, the algorithm has been tested with the change of the most important of them, i.e.: number of memory cells, number of clons, range of the Gaussian mutation and the crowding factor. The range of the changes of the particular parameters of the artificial immune system is presented in the Table 25. The results of the stage of the optimal parameters selection for particular mathematical functions are included in the Table 26.

Table 25. The range of the changes of the artificial immune system parameters

the number of memory cells	the number of the clones	crowding factor	Gaussian mutation
2, 4, 6, …, 100	2, 4, 6, …, 100	0,01; 0,02; …; 1.0	0,1; 0,2; …; 1.0

Table 26. The optimal parameters of the artificial immune system for particular functions

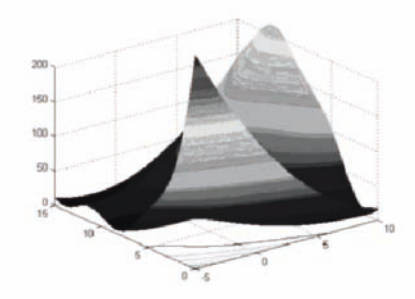
the number of memory cells	the number of the clones	crowding factor	Gaussian mutation
BRANIN			
2	2	0.48	0.1
GOLDSTEIN-PRICE			
12	2	0.45	0.5
RASTRIGIN			
2	4	0.45	0.4
GRIEWANGK			
2	2	0.45	0.1

a)

$$F(x)=\left(x_2-\frac{5}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(x_1)+10$$

$$-5\le x_1\le 10,\ \ 0\le x_2\le 15$$

$$\min\ F(x)=F(-\pi,12.275)=F(\pi,2.275)=$$
$$F(9.42478,2.475)=0.397887$$

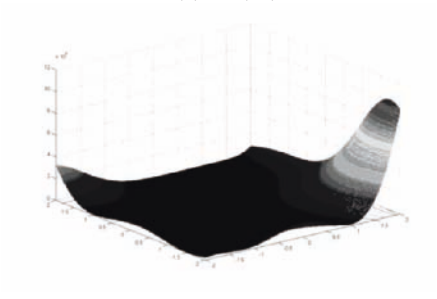


b)

$$F(x)=\left(1+(x_1+x_2+1)^2\left(19-14x_1+13x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right)$$
$$\left(30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2-48x_2-36x_1x_2+27x_2^2\right)\right)$$

$$-2\le x_i\le 2$$

$$\min F(x)=F(0,1)=3$$



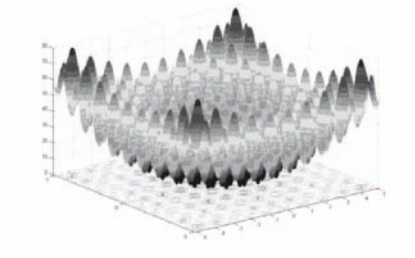
c)

$$F(x)=10n+\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)\right)$$

$$(-5.12\le x_i\le 5.12)$$

$$\min F(x)=F(0,0,\ldots,0)=0$$

for n =2



d)

$$F(x)=1+\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right)$$

$$(-600\le x_i\le 600)$$

$$\min F(x)=F(0,0,\ldots,0)=0$$

for n =2

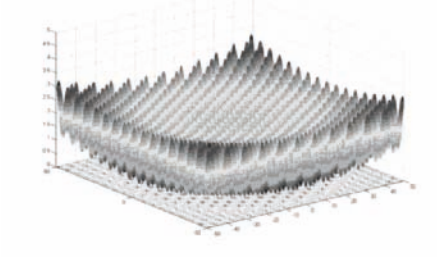


Figure 68. Tested mathematical functions: a) Branin function, b) Goldstein-Price function, c) Rastrigin function, d) Griewangk function

The sequential and distributed evolutionary algorithms applied for comparison with the artificial immune system uses evolutionary operators like the simple crossover and the Gaussian mutation. The selection is performed by the use of the ranking method. The optimal probabilities of the evolutionary parameters for particular mathematical functions are presented in the Table 27.

The result of the comparison between an artificial immune system and the sequential and distributed evolutionary algorithms are presented in the Table 28 and shown in the Figure 69.

The numbers of the objective function evaluations needed to the achievment of the value:
below 0.5 for the Branin function, below 3.1 for the Goldstein-Price function, below 0.1 for the Rastrigin function, below 0.1 for the Griewangk function have been compared.

Table 27. The optimal parameters of sequential and distributed evolutionary algorithms for particular functions

The number of subpopulations	The number of chromosomes in each subpopulation	The probability of simple crossover	The probability of Gaussian mutation
BRANIN			
1	20	100%	100%
2	10	100%	100%
GOLDSTEIN-PRICE			
1	20	100%	100%
3	7	100%	100%
RASTRIGIN			
1	20	100%	100%
2	10	100%	100%
GRIEWANGK			
1	10	100%	100%
2	5	100%	100%

Table 28. The result of the comparison between an artificial immune system and the sequential and distributed evolutionary algorithms – comparison of the average number objective function evaluations

Sequential EA	Distributed EA	artificial immune system
Average number objective function evaluations		
BRANIN		
188	155	171
GOLDSTEIN-PRICE		
287	188	325
RASTRIGIN		
9293	4659	7897
GRIEWANGK		
22285	13594	6019

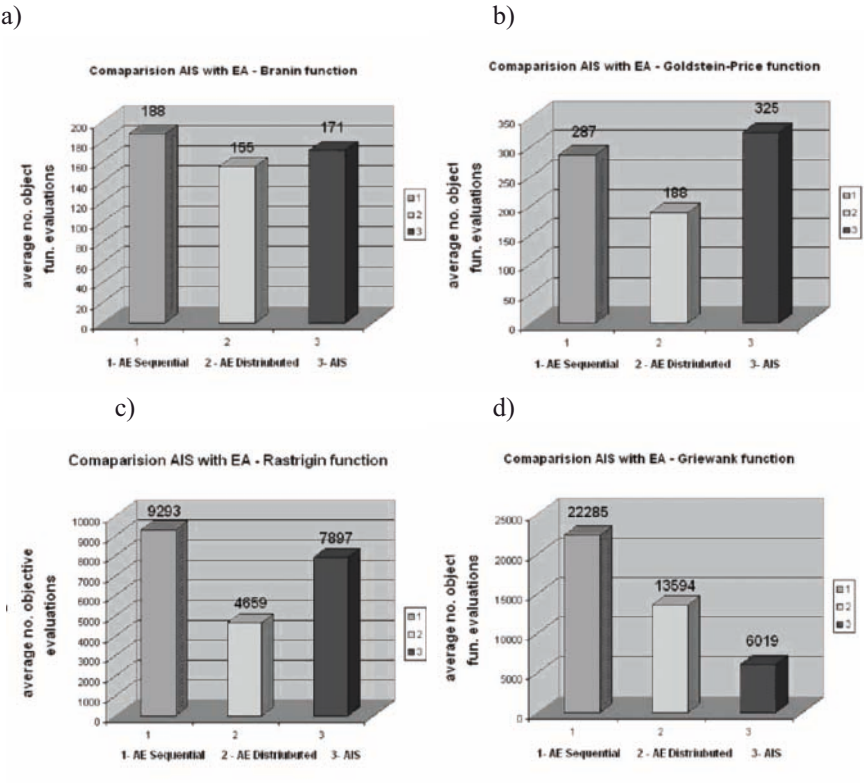


Figure 69. Comparison between AIS and EAs for: a) Branin function, b) Goldstein-Price function, c) Rastrigin function, d) Griewangk function

9.4 Topology immune optimization

Two numerical examples of immune optimization of topology structures:

- a plate in plane stress (*Example 14*),
- a solid body (*Example 15*),

by the minimization of the mass with imposed stress or displacement constraints are considered. The structures are considered in the framework of the theory of elasticity. The results of the examples are obtained by using an optimization method based on the artificial immune system with the parameters included in Table 29.

Table 29. The parameters of the artificial immune system

the number of memory cells	8
the number of the clones	4
crowding factor	25%
Gaussian mutation	20%

Example 14 – Immune optimization of a plate in plane stress. A rectangular 2-D structure (plane stress), of dimensions 100 × 200 mm, loaded with the concentrated force P in the centre of the lower boundary and fixed on the bottom corners is considered.

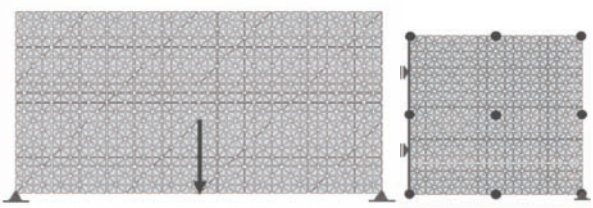


Figure 70. The plate (example 1); a) the geometry; b) the distribution of the control points of the interpolation surface

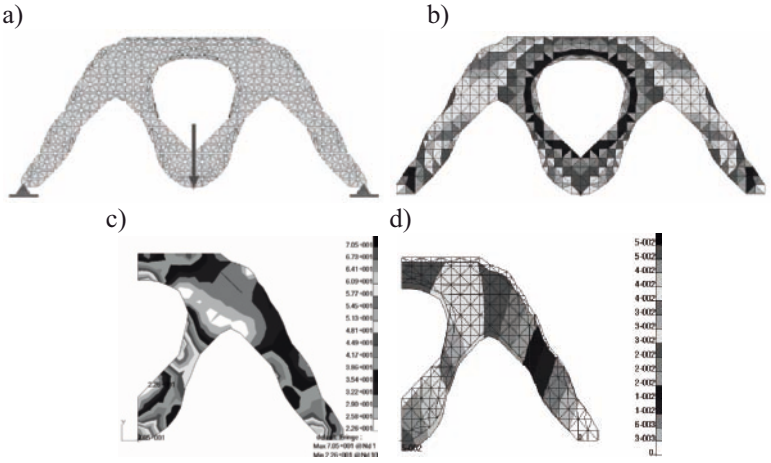


Figure 71. The results of the immune optimization of the plate: a) the solution of the optimization task; b) the map of mass densities; c) the map of stresses; d) the map of the displacement

Due to the symmetry a half of the structure has been analyzed. The input data and parameters of the artificial immune system are included in Table 30 and 29, respectively.

The geometry, the distribution of the control points of the interpolation surface are shown in the Figure 70a and 70b, respectively. The results of the optimization process are presented in the Figure 71.

Table 30. The input data to the optimization task of a shell bracket

σad [MPa]	the thickness [mm]	σmin ; p [MPa]	P1 [kN]	range of ρe [g/cm3]
80.0	4.0	1.0 ; 1.0	2.0	7.3 ≤ ρe < 7.5 elimination 7.5 ≤ ρe ≤ 7.86 existence

Example 15 – Immune optimization of a solid body. A 3-D structure with dimensions and loading is presented in the Figure 72a and 72b. The input data to the optimization program is included in Table 31.

The geometry, the distribution of the control points of the interpolation hyper surface are shown in the Figure 72c. The results of the optimization process are presented in the Figure 73 and Figure 74.

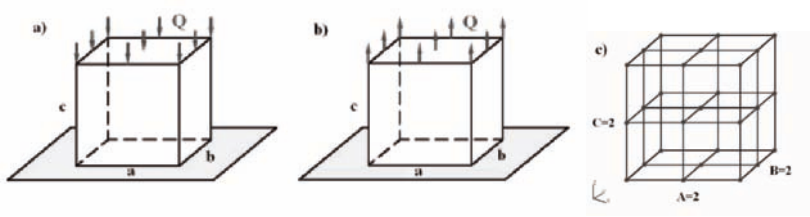


Figure 72. Two cases of loading with the hyper surface
a) first case (compression), b) second case (tension),
c) the distribution of the control points of the interpolation hyper surface

Table 31 Input data - geometry and loading

Dimensions [mm]			Loading Q	
a	b	c	compression	tension
100	100	100	-36.3 [KN]	36.3 [KN]

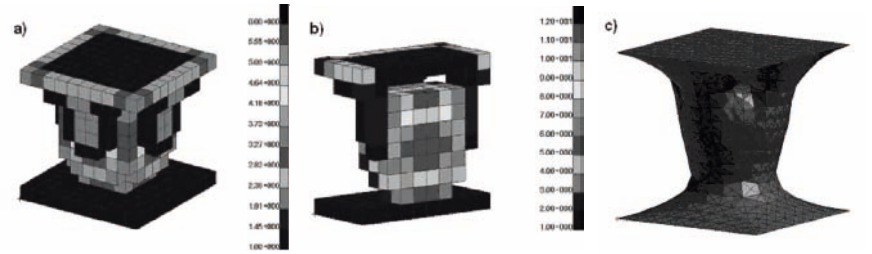


Figure 73. Distribution of mass density for the first case (compression) a), b) structure after 50 iteration (the best solution) c) structure after smooth

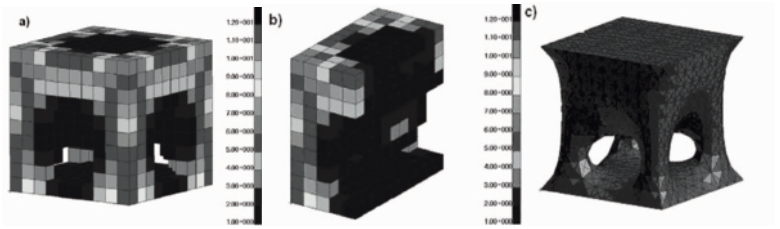


Figure 74. Distribution of mass density for the second case (tension) a), b) structure after 50 iteration (the best solution) c) structure after smooth

9.5 Concluding remarks

Taking into account the results of the comparison between the artificial immune system and the sequential and distributed evolutionary algorithms, performed on the basis of selected mathematical functions, one can conclude that the convergence to the global minimum depends on the optimization problem:

- for the Griewangk function the AIS is more effective than SEA and DEA,
- for the Goldstein-Price function the AIS is less effective than SEA and DEA,
- for the Branin and the Rastrigin functions the AIS is more effective than SEA but less than DEA.

Artificial immune systems can be considered as alternative to evolutionary algorithms intelligent global optimization techniques very useful in structural optimization.

Bibliography

Aleander, J.T. (2000). *An Indexed Bibliography of Distributed Genetic Algorithms*, University of Vaasa, Report 94-1-PARA, Vaasa, Finland.

Anagnostou, G. Rønquist, E. and Patera**,** A. (1992). A computational procedure for part design. *Computer Methods in Applied Mechanics and Engineering* 97:33-48.

António, C.A.C. and Douardo, N.M. (2002). Metal-forming optimization by inverse evolutionary search, *Journal of Material Processing Technology* 121: 403-413.

Arabas, J. (201). *Lectures on Evolutionary Algorithms*. WNT, Warszawa.

Augusto, O.B., Rabeau, S., Dépincé, Ph. and Bennis, F. (2006). Multi-objectivegenetic algorithms: A way to improve the convergence rate. *Engineering Applications of Artificial Intelligence* 19:501-510.

Badrinarayanan, S. (1997). Preform and die design problems in metalforming, *Ph.D. thesis*, Cornell University.

Beer, G. (1983). Finite element, boundary element and coupled analysis of unbounded problems in elastostastics. *Int. J. Numer. Meth. Eng.*, 19: 567-580.

Białecki, R.A., Burczyński, T., Długosz, A., Kuś, W. and Ostrowski, Z. (2005). Evolutionary shape optimization of thermolastic bodies exchanging heat by convection and radiation. *Computer Methods in Applied Mechanics and Engineering* 194:1839-1859.

Bendsøe, M. P. and Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering* 71: 197-224.

Bendsoe, M. P. and Sokołowski, J. (1988). Design sensitivity analysis of elastic-plastic problems. *Mechanics of Structures and Machines* 16: 81-102.

Brebbia, C.A., Telles, J.C.F. and Wrobel, L.C (1984). *Boundary Element Techniques*. Springer –Verlag, Berlin.

Burczyński, T. (1995). *The Boundary Element Method in Mechanics*. WNT, Warsaw.

Burczyński, T. and Długosz, A. (2002). Evolutionary optimization in thermoelastic problems using boundary element method. *Computational Mechanics*, 28: 317-324.

Burczyński, T., Długosz, A. and Kuś, W. (2006). Parallel evolutionary algorithms in shape optimization of heat radiators. *Journal of Theoretical and Applied Mechanics* 44: 351-366.

Burczyński, T. and Kokot, G. (1998). Topology optimization using boundary elements and genetic algorithms. Proc. *Fourth Congress on Computational Mechanics, New Trends and Applications*, (eds. Idelsohn S.R., Oñate E., Dvorkin E.N.), Barcelona, CD-rom.

Burczyński, T., and Kuś, W. (2001). Application of evolutionary algorithms in optimal design of elasto-plastic structures. *Computer Methods in Materials Science* 1:189-195.

Burczyński, T. and Kuś, W. (2003). Optimal design of elasto-plastic structures using sequential and distributed evolutionary algorithms. *Chapter 3 in Information Technology in Process Engineering of Metals* (Eds. A. Piela, F. Grosman,J. Kusiak, M.Pietrzyk), Gliwice, 108-142.

Burczyński, T. and Kuś, W. (2004). Optimization of structures using distributed and parallel evolutionary algorithms. *Parallel Processing and Applied Mathematics, Lecture Notes on Computational Sciences 3019*, Springer, 572-579.

Burczyński, T., Kuś, W., Długosz, A. and Orantek, P. (2004). Optimization and identification using distributed evolutionary algorithms, *Engineering Applications of Artificial Intelligence* 17: 337-344.

Burczyński, T. and Osyczka, A. (eds) (2004). *Evolutionary Methods in Mechanics*. Kluwer, Dordrecht.

Burczyński, T., Poteralski, A. and Szczepanik, M. (2003). Genetic generation of 2-D and 3-D structures. In: *Computational Fluid and Solid Mechanics 2003* (ed. K.J. Bathe), Vol. 2, Proc. *Second M.I.T. Conference on Computational Fluid and Solid Mechanics Institute of Technology*, Cambridge, Massachusetts 02139 U.S.A. Elsevier, 2221-2225.

Burczyński, T., Poteralski, A. and Szczepanik, M. (2007). Topological evolutionary computing in the optimal design of 2D and 3D structures. *Engineering Optimization* 39:811-830.

Cantu-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis* 10: 141-171.

Cantu-Paz, E. (1999). Migration policies, selection pressure, and parallel evolutionary algorithms. In Brave S., Wu. A. (Eds), *Late Breaking Papers at the Genetic and Evolutionary Computation Conference. Orlando* FL.

Cantu-Paz, E. (2000). On the effect of migration on the fitness distribution of parallel evolutionary algorithms. In *Workshop on Evolutionary Computation and Parallel Processing at GECCO-2000*, Las Vegas, NV, 3-6.

Castro, L.N. and J. Timmis (2003) Artificial immune systems as a novel soft computing paradigm. *Soft Computing* 7(8): 526-544.

Coello, C.A. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*. 1(3): 129-156.

Coello, C.A. and Christiansen, A.D. (2000). Multiobjective optimization of trusses using genetic algorithms. *Computers and Structures* 75:647-660.

Chapman, C., Saitou, K. and Jakiela, M. (1995): Genetic algorithms as an approach to configuration and topology design. *ASME Journal of Mechanical Design* 45.

Deb, K. (1999). Multi-objective genetic algorithms: problem difficulties and construction of test problems. *Evolutionary Computation* 7(3): 205-230.

Długosz, A. (2004). Evolutionary computation in thermoelastic problems. In: *Evolutionary Methods in Mechanics* (eds. T. Burczyński and A. Osyczka). Kluwer, Dordrecht, 69-80.

Długosz, A., (2001). *Boundary element method in sensitivity analysis and optimisation of thermoelastic structures*, PhD thesis, Silesian University of Technology, Gliwice.

Eschenauer, H.A. and Schumacher, A. (1995). Simultaneous shape and topology optimization of structures. Proc. *First World Congress of Structural and Multidisciplinary Optimization*, (eds. Olhoff N., Rozvany G.I.N.) Pergamon, Oxford, 177-184.

Fonseca, C. M. and Fleming, P.J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation* 3(1): 1-16.

Jensen, E. (1992). *Topological Structural Design Using Genetic Algorithms*. Doctor of Philosophy Thesis, Purdue University.

Kirkpatrick, S., Gelatt, C. and Vecchi, M. (1983). Optimization by simulated annealing. *Science* 220: 671-680.

Kuś, W. (2002). *Coupled Boundary and Finite Element Methods in Optimization of Mechanical Structures*. Ph.D. Thesis, Gliwice.

Kuś, W. and Burczyński, T. (2004). Distributed evolutionary algorithms in optimization of nonlinear solids. In: *Evolutionary Methods in Mechanics,* (eds. T. Burczyński, A. Osyczka). Kluwer, Dordrecht, 229-240.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolutionary Algorithms*. Springer-Verlag, Berlin.

MSC.MARC (2001). Theory and user information Vol. A-D, MSC Software Corporation.

Piegl, L. and Tiller W. (1997). *The NURBS Book.* Springer- Verlag. Berlin.

Sandgren, E., Jensen, E. and Welton, J. (1990). Topological design of structural components using genetic optimizastion methods, Proc. *Winter annual meeting of the American Society of Mechanical Engineers*, Dallas, Texas, 31-43.

Shewchuk, J. R. (1996). Triangle: engineering a 2D quality mesh generator and Delaunay triangulator, *First Workshop on Applied Computational Geometry, Association for Computing Machinery*, Philadelphia, Pennsylvania, USA. 124-133.

Szczepanik, M. (2003). *Optimization of 2-D Structures Using Evolutionary Algorithms*. Ph. D. Thesis, Silesian University of Technology, Gliwice.

Tanese, R. (1989). Distributed genetic algorithms. *Proc. 3rd ICGA* (ed. J.D. Schaffer),. San Mateo, USA, 434-439.

Xie, Y.M. and Steven, G.P. (1997). *Evolutionary Structural Optimization.* Springer, London.

Wierzchoń, S.T. (2001). *Artificial Immune Systems: Theory and Applications*. EXIT, Warsaw.

Woon, S.Y., Tong, L., Querin, O.M. and Steven, G.P. (2003). Optimising topologies through a Multi-GA System. *Proc. 5th World Congress on Structural and Multidisciplinary Optimization WCSMO 2003*, Italy, Venice.

Zabaras, N. Bao, Y., Srikanth, A. and Frazier, W. G. (2000). A continuum Lagrangian sensitivity analysis for metal forming processes with applications to die design problems, *Int. J. Numer. Meth. Engng* 48: 679 –720.

Zhao, X., Zhao, G., Wang, G. and Wang, T. (2002). Preform die shape design for uniformity of deformation in forging based on preform sensitivity analysis. *Journal of Material Processing Technology* 128: 25-32.

Zienkiewicz, O.C. and Taylor, R.L. (2000). *The Finite Element Method, Solid Mechanics*. Butterworth, Oxford.

CHAPTER 3

Applications of GA and GP to Industrial Design Optimization and Inverse Problems

V.V. Toropov[1,2], L.F. Alvarez, O.M. Querin[2]

[1]School of Civil Engineering,
[2]School of Mechanical Engineering, University of Leeds, UK

Abstract. In this chapter the use of Genetic Algorithms and Genetic Programming for various industrial problems is discussed. Particular attention is paid to the case of difficult design optimization problems in which either (or both) (*i*) response functions are computationally expensive as well as affected by numerical noise and (*ii*) design variables are defined on a set of discrete variables.

1 Introduction

Many real-life design optimization problems have the following common features that make application of well-established gradient-based optimization algorithms difficult, if not impossible:

- the objective and constraint functions are evaluated as a result of expensive numerical computations, e.g. using an FEM or a CFD code;
- function values and/or their derivatives may contain numerical noise;
- domain-dependent calculability of response functions, i.e. situations when these functions cannot be evaluated at some points of the design variable space;
- design variables are defined on a set of discrete values.

The first three features also arise in industrial problems of stochastic analysis, particularly when a Monte Carlo simulation is performed. Hence the only realistic way of addressing such problems is to build high quality metamodels (also referred to as surrogate models or global approximations) that (i) describe the behaviour of the system (or a process) with a sufficient accuracy as compared to simulation, (ii) are computationally inexpensive and (iii) do not possess any significant amount of numerical noise. When such metamodels are obtained and their quality verified, it becomes possible to use them in optimization or a Monte Carlo simulation in lieu of expensive simulations.

The last feature mentioned above, the discrete nature of design variables, is most typically addressed by the use of suitable optimization techniques, of which

a Genetic Algorithm is currently the most popular choice. Such an algorithm may be calling a numerical simulation directly (when it is feasible) or used in conjunction with a metamodel that has been built previously. Sections 2 and 3 show applications of such techniques.

In order to build a metamodel it is necessary to perform sampling by running a sufficient number of numerical simulations in the space of optimization variables and/or stochastic factors according to a carefully chosen design of experiments (DOE). Experiments usually mean numerical simulation at various combinations of optimization or stochastic variables but may also include laboratory or in situ experiments. Uniform designs of experiments that are based on the concept of an optimal Latin hypercube have been developed with the intention of gathering as much information on the behaviour of the system (or a process) as possible with a minimum number of sampling points. These include nested (i.e. individual uniform DOEs used separately for metamodel building and validation and then can be merged while remaining uniform) and extended (to include pre-defined sampling points while preserving uniformity of a DOE). Section 4 describes the use of permutation GA for the creation of uniform DOEs.

The most typically used metamodelling techniques include:

- Response surface methodology that is further divided into the following classes:
  - Linear (e.g. polynomial) regression
  - Nonlinear regression
  - Mechanistic models
  - Application of the Genetic Programming methodology for selection of the structure of an analytical expression treated as a metamodel. This has been used for creating analytical descriptions of various engineering systems and processes, also using data produced by laboratory experimentation, Toropov (2001).
- Artificial neural networks
- Radial basis functions
- Kriging, Matheron (1963)
- Multivariate Adaptive Regression Splines (MARS), Freedman (1991)
- Metamodelling based on the interaction of low- and high-fidelity simulation models that is beneficial for creating high quality metamodels when a lower complexity simulation model is available in addition to the original high fidelity simulation model. In such a case the metamodel can be based on that low fidelity model which is corrected and tuned using only a small number of runs of the high fidelity model, Toropov (2001)
- Moving Lest Squares Method (MLSM), Choi et al. (2001), Toropov et al. (2005), is a global metamodelling technique that is flexible enough to describe the sampling data with high accuracy when the amount of

numerical noise is small (in such case it is similar to kriging) but also has a capability of filtering out numerical noise when it becomes an issue.

Once the metamodel has been obtained and verified, various optimization techniques can be used, these include genetic algorithms (particularly when design variables are defined on a set of discrete variables) and gradient-based optimization methods. In the case of stochastic analysis, a large Monte Carlo simulation can be performed in order to assess reliability and robustness of a system or a process.

Sections 5, 6 and 7 show applications of Genetic Programming to metamodel building.

2 Weight Optimization of a Formula One Car Composite Component Using Genetic Algortithm

In the development of a new Formula One car, the design process takes approximately four months and has to go through as many iterations as possible. In such a competitive environment small improvements can be crucial, and the use of a robust technology to produce the best possible design is an advantage.

Following a review of industry practice (Nevey and Stephens, 2000), there appears to be no consistent commercial approach for the application of optimization techniques to composite laminates.

A genetic algorithm (GA) has been applied to search for the optimum combination of discrete design variables (fibre orientation and number of plies) that produce the maximum structural stiffness at the lowest mass. Simulations of the composite components have been performed using the linear static analysis code Altair OptiStruct (2000).

An application is presented where the developed methodology is applied to the weight optimization of the Jaguar R3 front wing, Stephens et al. (2002).

2.1 Optimization problem

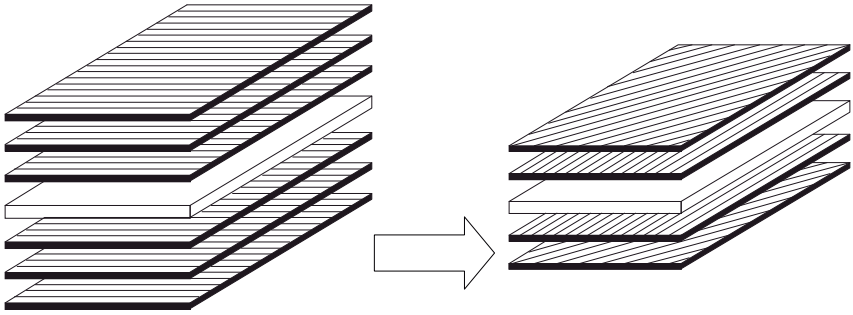
The definition of a directional fibrous laminated composite requires the specification of the fibre direction and the number of plies, see Figure 1. The ultimate objective of the optimization methodology is to automatically determine the optimum laminate configuration (e.g. the optimum number of plies and the orientation of every ply).

The optimization problem is to minimize the weight $F_0(\boldsymbol{x})$ of a laminated composite component, subject to various constraints $F_j(\boldsymbol{x})$. The design variables $\boldsymbol{x}$ are the orientation of the fibre and the number of plies, which have been grouped into bundles for manufacturing requirements:

$$F_0(\boldsymbol{x}) \to \min, \quad F_j(\boldsymbol{x}) \le 1 \ \ (j = 1,..., M),$$
$$A_i \le x_i \le B_i \ \ (i = 1,..., N) \tag{1}$$

2.2 Genetic algorithm

A genetic algorithm (Goldberg, 1989) is a machine learning technique modelled upon the natural process of evolution. Genetic algorithms differ from conventional optimization techniques in that they work on a whole population of individual objects of finite length, typically binary strings (chromosomes), which encode candidate solutions using a problem-specific representation scheme. These strings are decoded and evaluated for their fitness, which is a measure of how good a particular solution is. Following Darwin's principle of survival of the fittest, strings with higher fitness values have a higher probability of being selected for mating purposes to produce the next generation of candidate solutions.



Baseline configuration Optimized lay-up

Figure 1. Optimization of a composite laminate

Selected individual design solutions are reproduced through the application of genetic operators. A string selected for mating is paired with another string and with a certain probability each pair of parents undergo crossover (sexual recombination) and mutation. The strings that result from this process, the children, become members of the next generation of candidate solutions.

This process is repeated for many generations in order to artificially evolve a population of strings that yield a solution to a given problem, see Figure 2. For full detail of GA theory applied to the optimization of composites, see Gürdal et al. (1999).

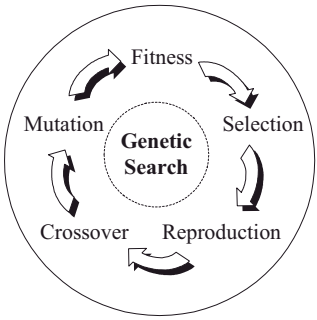


Figure 2. Evolutionary algorithm

The main features of the GA used in this study are:

- A small percentage of the population, the *elite*, is transferred unchanged to the next generation.
- *Uniform crossover* (Figure 3) creates an offspring by copying bits from the corresponding parents selected randomly and with equal probability.

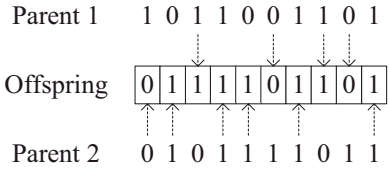


Figure 3. Uniform crossover

- Tournament selection of size two (or more) selects the best individual from a subpopulation of two (or more) randomly selected strings.
- Linear scaling of the fitness function applies the transformation

$$F_{scaled} = a * F + b$$

where F is the actual fitness of an individual and a and b are derived from the coordinate system change $[F_{av}, F_{max}]$ to $[F_{av}, c*F_{max}]$ as follows:

$$a = \frac{(c-1)\ F_{av}}{F_{max} - F_{av}} \qquad b = (1-a)\ F_{av}$$

where, for a current population, F_{av} is the average fitness value and F_{max} is the fitness value of the best individual. In this study, c was taken as 3. If

F_{scaled} goes negative, the corresponding individual is repeatedly replaced with another one created randomly until F_{scaled} becomes positive.

- *Gray coding* is a binary string representation such that 2 consecutive numbers differ in only 1 bit (Figure 4).

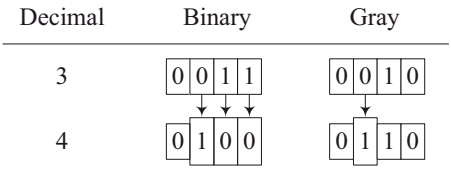


Figure 4. Gray coding

This coding is less disruptive than binary coding for the mutation operator and makes the search more incremental.

Assuming G is a Gray coded string with n bits $[G_n\ G_{n-1}\ \ldots\ G_1]$ where G_n is the most significant bit, B is the corresponding binary coded string $[B_n\ B_{n-1}\ \ldots\ B_1]$, and ^ is the bitwise exclusive OR operator (XOR), the conversion algorithms are:

- From Gray to binary:

$$j = n; \qquad B_j = G_j$$
$$j = n\text{-}1 \rightarrow 1; \qquad B_j = B_{j+1} \wedge G_j$$

- From binary to Gray:

$$j = n; \qquad G_j = B_j$$
$$j = n\text{-}1 \rightarrow 1; \qquad G_j = B_{j+1} \wedge B_j$$

- Local search (Sahab, 2001) performs a coordinate search in the positive and negative direction through all the variables (Figure 5). In this search:
  - Initial starting point is the optimum solution returned from the GA.
  - The increment steps Dx are those defined for the corresponding discrete variables. Steps are increased or decreased within the variable range.
  - Every new solution serves as starting guess for another local search, until no improvement can be found.
- If two strings have the same fitness, one of them is deleted and a new one randomly created.

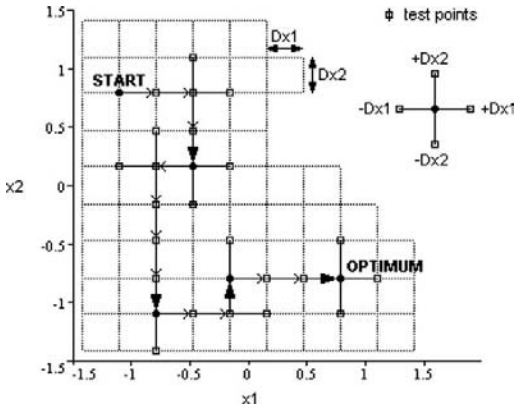


Figure 5. Local search

For constrained optimization problems, an exterior penalty function can be adopted to transform a constrained optimization problem into an unconstrained one. Penalty functions are defined in the exterior of the feasible domain, so that constraints are applied only when they are violated. The fitness function is defined as follows:

$$\text{Fitness} = \overline{F}_0(\boldsymbol{x}) + r \sum_{i=1}^{M} max\left[\left(F_j(\boldsymbol{x}) - 1 \right)^t, 0 \right] \tag{2}$$

where $\overline{F}_0$ is the normalised objective function, r is a penalty multiplier and t defines the power of the penalty function.

GA parameters

The following parameters have been assumed in the optimization process:

- Population size: 200
- Elite: 10%
- Mutation: 1%
- 50% of the population with the worst fitness are replaced in every generation.
- Gray coding with vacant positions (if available) filled with bits from the beginning of the string.
- In the expression (2), penalty multiplier r = 10 and linear penalty function (t = 1) lead to a moderate penalisation which sometimes allows for unfeasible solutions. A quadratic penalty functions (t = 2) would typi-

cally converge to an unfeasible solution, while a square root penalty function ($t = 0.5$) would be a conservative approach and converge in the feasible domain.

2.3 Front wing optimization

The final objective of this study was to optimise the weight of the front wing in the Jaguar R3 Formula One car for the 2002 racing season. The baseline model was divided into 5 designable areas with 3 bundles of plies in each section, see Figure 6. This lay-up defines 15 fibre angles and 15 numbers of plies. The allocation of design variables assumed that the model is symmetric about the centreline of the wing.

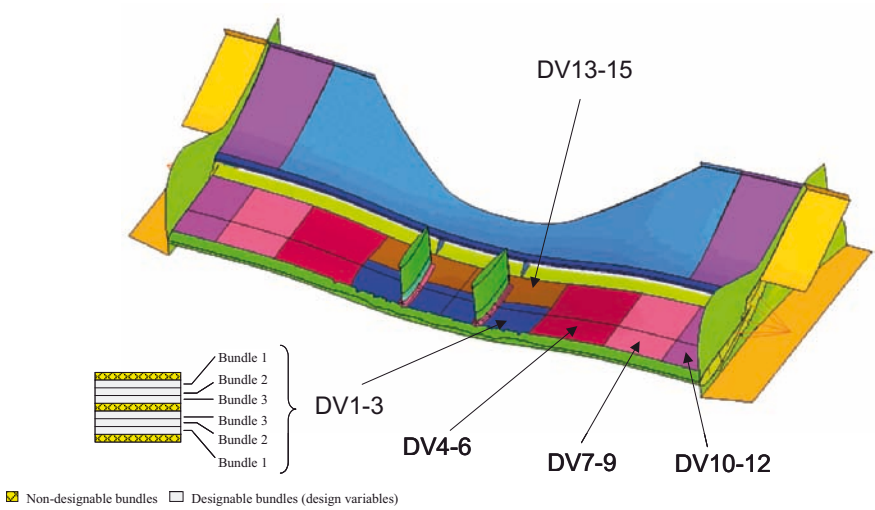


Figure 6. R3 front wing lay-up

The problem was to minimize the mass of the wing, subject to the following normalised constraints:

- Maximum displacement under a 50 kg mass placed on the wing (FIA regulation, FIA, 2002) ≤ 1 mm.
- Maximum displacement under aerodynamic loading (calculated from a CFD analysis) ≤ 1 mm.

- Twist under aerodynamic loading (relative displacement of the leading and trailing edge) ≤ 1 mm.

The optimization was carried out for both fibre orientation and number of plies concurrently. The proposed methodology achieved a 5% saving over the baseline weight of the wing. The GA convergence history is plotted in Figure 7.

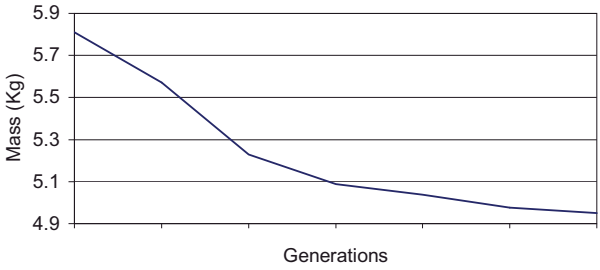


Figure 7. GA convergence history

2.3 Conclusion

A methodology for optimising composite lay-ups combines a genetic algorithm with a Finite Element software and finds concurrently the fibre orientation and the number of plies.

The results of the optimization have been used in the following ways:

- It showed trends of the wing lay-up, i.e. biased more to the reduction of bending in the middle of the wing and biased more to the reduction of twist at the outer edge.
- The final results produced the wing lay-up that was put into the front wing for the R3 Formula One car.

3 Application of Optimization Techniques to Structural Damage Recognition

An identification of the location and degree of damage sustained by an engineering structure is often of considerable importance in structural engineering practice. Visual inspection and extensive field testing are usually employed to locate and quantify the degree of degradation of a structure but this can be expensive and time consuming. An alternative method of non-destructive testing is based on the monitoring of changes in dynamic structural characteristics such as natural frequencies. These can be obtained by measurements at a limited number of points of the structure (often even one point) and are relatively independent of the chosen location. Most importantly, these characteristics are sensitive to changes in the mass and

stiffness of the structure. Therefore, data on the changes in the dynamic characteristics can be used to calculate local decrease in the stiffness which indicates the presence of the structural damage. The problem then can be formulated as an identification problem of finding the stiffness matrix which reproduces the measured data, see, e.g., Hassiotis and Jeong (1993), and can be solved using advanced optimization techniques.

Various formulations have been used by investigators in the past for the adjustment of the stiffness matrix to meet natural frequencies and mode shapes observed in the experiment, see Baruch (1982), Kabe (1985) and Lapierre and Ostiguy (1990). Sensitivity analysis techniques that make use of the sensitivities of the eigenvalues and eigenvectors to the changes in the stiffness have also been used to verify and upgrade the analytical model given measured frequencies and eigenmodes. Cawley and Adams (1979) used the first order perturbation of the basic eigenvalue equation to obtain sensitivities necessary to locate the damage in a structure using natural frequencies. The location of the damage was assumed to be where the theoretically determined ratio of changes in any two frequencies was equal to the experimentally measured value. Hajela and Soeiro (1990a) presented a more general approach based on the use of either static displacements or frequencies and modes for damage detection. The damage was modelled on an element-by-element basis as changes in sectional properties, which then contribute to variation in the elements of the structural stiffness matrix. The output error approach, where changes are made in numerical model to match the experimental response, and the equation error approach where the model parameters are adjusted to obtain a match between the left and right hand sides of the response equation, have been used to detect a simulated damage in a series of truss and semimonocoque structures (see Hajela and Soeiro, 1990b).

In this section the output error method of system identification is used to demonstrate the assessment of the presence of damage in steel frame structures. The approach is based on the use of measurements of natural frequencies only because information on shape modes is not easily obtainable with sufficient accuracy. The results obtained using a derivative-based optimization technique are compared to those obtained using a genetic algorithm.

3.1 Identification problem formulation

In a finite element formulation characteristics of the structure are defined in terms of the stiffness and mass matrices **K** and **M**, respectively. Any variations in these matrices, e.g. introduced by damage, would affect the dynamic response characteristics of the structure. The analytical model describing the eigenvalue problem for an undamped system can be stated in terms of the matrices **K** and **M**, the *i*-th eigenvalue ω_i^2, and the corresponding eigenmode ξ_i as follows:

$$(\mathbf{K} - \omega_i^2 \mathbf{M})\, \xi_I = 0 \tag{3}$$

Matrices **K** and **M** are to be adjusted to minimize the differences between the experimentally observed eigenvalues and values obtained from the analytical (e.g., finite element) model. As a variation in the system matrices results in changed frequency response, the damage assessment problem, formulated as an inverse problem, is to relate these differences to changes in specific elements of the system matrices. In order to describe the influence of the presence and extent of damage on the matrices **K** and **M**, the optimization variables ***x*** are to be introduced such as sectional properties of individual structural elements (the cross-sectional area, moments of inertia) or material parameters (Young's modulus, etc.). These dependencies may be stated in a functional form: $\mathbf{K} = \mathbf{K}(\mathbf{x})$, $\mathbf{M} = \mathbf{M}(\mathbf{x})$.

Two basic approaches can be suggested for the description of the presence, location and the extent of structural damage by optimization variables ***x***. In the first one, an individual variable x_i can describe the extent of possible damage at *i*-th location, e.g. in the *i*-th finite element. This formulation leads to a continuous optimization problem, can easily describe the presence of multiple damage but the number of variables can be large when a large scale finite element model is used.

Alternatively, the vector of optimization variables can be considered as a set of *L* couples $x_1^1, x_2^1, x_1^2, x_2^2, \ldots, x_1^L, x_2^L$ where x_1^j is a number of a damaged element in the FE model and x_2^j describes the extent of damage occurring in it, $j = 1,\ldots, L$ and *L* is the assumed maximum number of damaged elements. Such approach leads to a considerably smaller number of variables but presents a discrete or mixed discrete-continuous optimization problem. In this section the use of both approaches is presented.

Using the output error approach, the damage identification problem can be formulated as the following optimization problem:

Find the optimization variables ***x*** by minimizing the differences between the frequencies ω_i^M measured in the course of laboratory experiment or operation and the frequencies $\omega_i^A(\mathbf{x})$ obtained by the Finite Element analysis:

$$\text{minimize } \left[\omega_i^M - \omega_i^A(\mathbf{x})\right]^2,\ i = 1,\ldots,F \tag{4}$$

where *F* is total number of modes of vibration used for the identification. The formulated problem is a multicriterion one but it can be transformed to a more conventional optimization problem by formulating a single criterion. A linear combination of individual differences (4) is a most typically used one, the optimization problem can then be reformulated in the following form:

$$\text{minimize } \sum_{i=1}^{F} w_i \left\{\left[\omega_i^M - \omega_i^A(\mathbf{x})\right] / \omega_i^M\right\}^2 \tag{5}$$

where the weights w_i describe the relative importance of the match between frequencies of the *i*-th mode.

3.2 Experimental procedure

The test structure used for the investigation was a steel portal frame clamped at the base of both columns as shown in Figure 8. All parts of the frame have the same 800 mm × 400 mm rectangular hollow section of 4 mm thickness. The first ten natural frequencies were measured on the undamaged frame and also when three stages of progressive damage (classified as mild, medium and severe) were applied at the location close to the top joint. In all cases, the damage was applied by removing the material symmetrically relative to the beam's neutral axis thus reducing the cross section area to 64%, 54% and 35% of the original value for the undamaged structure.

For the experimental data acquisition a standard technique of modal structural testing has been used. Natural frequencies were measured by the impulse technique because of its speed and ease of execution. The oscillations in the structure have been excited with an instrumented hammer with a build-in force transducer Bruel & Kjaer (B&K) type 8200. The acceleration of frame was measured by using a 14 g accelerometer (B&K type 4369) so the weight of that comparing to the weight of the frame was negligible. The signals from hammer and accelerometer are amplified by B&K charge amplifiers type 2635 as schematically shown in Figure 8.

The excitation and response signals have been measured and processed using the dual channel spectral analyser B&K type 2032. It transforms the two sampled time functions into frequency spectra by a Fast Fourier Transformation (FFT) and subsequently computes the ratio of these functions yielding the Frequency Response Function (FRF). For example, two measured FRFs for the undamaged frame (solid curve) and the damaged frame (dashed curve) are shown in Figure 9. The difference between the natural frequencies for the damaged and the undamaged frame can easily be seen. As expected, the natural frequency for the damaged structure is

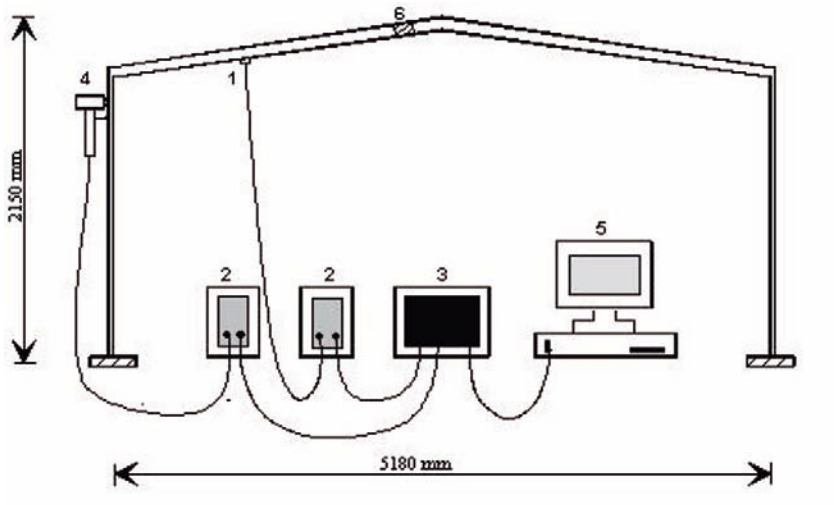


Figure 8. The portal frame and instrumentation set up: accelerometer (1), charge amplifier (2), dual channel analyzer (3), instrumented hammer (4), PC (5), damage position (6)

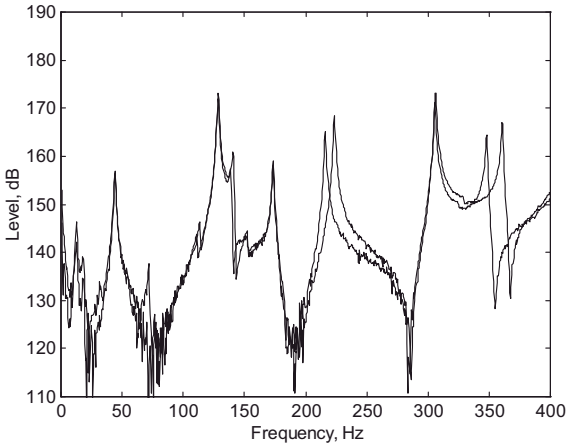


Figure 9. Plot of frequency response functions for undamaged (solid curves) and damaged frame (dashed curves)

lower than for the undamaged one. The adequate number of the accelerometer positions along the perimeter of the frame has been established to ensure that no

resonance is overlooked. To obtain a sufficient resolution in low frequencies the measurements have been repeated in several frequency spans (25, 50, 100, 200 and 400 Hz.). Some additional measurements were executed to detect and eliminate out of plane mode shapes and frequencies. The results are shown in Table 1.

Table 1. Experimental and analytical values of natural frequencies

Mode number	Undamaged frame					Damaged frame			
	Experiment	Analytical (FEM)				Experimental			
		Before validation		After validation		Mild damage		Medium damage	
	Freq., Hz	Freq., Hz	Difference, %	Freq., Hz	Difference, %	Freq., Hz	Change, %	Freq., Hz	Change, %
1	12.59	15.41	22.3	12.61	0.1	12.63	0.3	12.59	0.0
2	18.47	19.59	6.0	18.31	-0.8	18.34	-0.7	18.13	-1.9
3	44.13	45.34	2.7	43.65	-1.0	44.13	0.0	44.13	0.0
4	76.38	76.50	0.1	75.47	-1.1	74.38	-2.6	72.13	-5.6
5	128.5	135.0	5.0	128.1	-0.3	128.5	0.0	128.5	0.0
6	140.8	163.7	16.3	141.0	0.1	140.0	-0.5	139.0	-1.2
7	173.8	198.9	14.5	175.0	0.7	173.8	0.0	173.5	-0.1
8	223.5	237.0	6.0	225.2	0.7	220.0	-1.6	216.0	-3.4
9	306.5	313.7	2.3	308.4	0.6	306.0	-0.2	305.5	-0.3
10	361.0	367.3	1.7	364.7	1.0	354.0	-1.9	348.0	-3.6

A finite element model of 56 plane beam elements was used. It was found to be very important to validate the model by minimizing the difference between the experimental and analytical results. An optimization procedure was used to validate the model using the experimental results on ten first natural frequencies for the undamaged frame. Four parameters have then initially been considered as optimization variables: the Young's modulus and the density of the material, the area of cross section, and the moment of inertia of small artificially introduced elements at the base of both columns. Variation in the last parameter was intended to cover the uncertainty of the boundary conditions (clamped columns) and had the most profound effect on the validation. In addition, the effect in changes in the FE mesh was studied and found to be insignificant. The results of the model validation are presented in Table 1

3.3 Application of a derivative-based technique

The continuous optimization problem has been solved by the Sequential Quadratic Programming (SQP) method combined with a genetic algorithm used to produce a high quality starting guess.

First it was assumed that there was only one damage occurring at the one of the joints, as these are more probable points in the structure to be damaged (see Ravaii et al., 1998a). Then there are just five possible places for damage location and because of the symmetry condition only three optimization variables were defined in the optimization problem: one for the top joint, one next to the corner and one at the base. Both the location and size of the damage were successfully detected. In order to determine the number of modes necessary to reliably detect the damage, the number of modes was incremented one by one. The results are presented in Table 2.

Table 2. Damage detection and the number of modes used (three possible locations)

Type of actual damage	Damage location and extent		Number of frequencies used for damage identification and corresponding percentage of remaining area of cross-section found									
	Joint	Area (%)	1	2	3	4	5	6	7	8	9	10
Mild	1	100	81	105	105	105	105	105	103	90	89	95
	2	64	77	99	99	67	67	66	66	63	63	59
	3	100	102	102	104	103	103	102	100	104	105	104
Medium	1	100	86	70	105	105	105	105	95	89	86	98
	2	54	26	98	53	50	50	50	50	49	49	47
	3	100	93	103	95	96	97	97	96	97	101	95
Severe	1	100	86	105	105	105	90	88	89	104	105	103
	2	35	26	33	33	32	33	33	33	32	32	32
	3	100	93	92	93	95	105	105	105	104	105	103

As could be expected, for a mild damage at least first four natural frequencies were needed to detect the damage but for the medium and severe damage the first three and two modes, respectively, were sufficient. Next, it was assumed that the damage could happen at a greater number of possible locations (see Ravaii et al., 1998b). Thus eight more possible locations were considered: five additional possible points of damage on a rafter and three other ones on a column and also three points at the joints as in the previous formulation. There was no restriction on the number of damaged elements applied, i.e. this could vary from zero (no damage found) to 11 (damage at all possible locations). The first six natural frequencies were used to detect the damage. The location and size of the damage were successfully detected again. The results are presented in Table 3.

Table 3. Damage detection using the first six natural frequencies (eleven possible locations)

Type of damage		Damage location and extent										
	FE. No.	1	2	3	4	5	6	7	8	9	10	11
Mild	Exact area (%)	100	64	100	100	100	100	100	100	100	100	100
	Detected area (%)	105	65	105	105	93	101	105	105	105	98	101
Medium	Exact area (%)	100	54	100	100	100	100	100	100	100	100	100
	Detected area (%)	105	50	100	105	96	105	105	105	105	95	101
Severe	Exact area (%)	100	35	100	100	100	100	100	100	100	100	100
	Detected area (%)	100	32	102	105	96	98	105	105	105	93	95

3.4 Application of a genetic algorithm

A genetic algorithm has been applied to the damage recognition problems in both continuous and discrete-continuous formulations.

First, in order to compare the performance of the GA with that of SQP, a GA has been applied to the continuous optimization problem. The cross-sectional areas of eight elements of the FE model were considered as optimization variables. Lower and upper bounds of these cross-sectional areas were taken as 1 and 128 respectively, where 100 (or near) represents an undamaged element. The discretization of design variables was defined by increments by one thus resulting in the overall string length of 56 for all of the eight design variables. The following parameters of the genetic algorithm have been used: size of the population 60, proportion of the elite of 0.4, the probabilities of crossover and mutation have been taken as 0.6 and 0.01 respectively. The computations were carried out for three, five and eight possible damaged locations and in all cases the damage was successfully detected. The results for damage detection with eight possible damage points (eight optimization variables) are shown in Table 4.

Table 4. Damage detection using GA (eight possible locations)

	Damage location and extent							
Element No.	1	2	3	4	5	6	7	8
Exact area (%)	100	64	100	100	100	100	100	100
Detected area	86	68	96	100	84	95	89	104

Next, a GA has been applied to the discrete-continuous optimization problem (second formulation). In a discrete-continuous optimization problem, as described above, the vector of variables is presented as a set of L couples $x_1^1, x_2^1, x_1^2, x_2^2, \ldots, x_1^L, x_2^L$ where L is the assumed maximum number of dam aged elements. In each couple x_1^j is a number of a damaged element describing the location of damage (a discrete variable) and x_2^j was accepted as a number between 1 and 128 describing the extent of damage occurring at a corresponding j-th location (a continuous but discretized variable). Such approach leads to a considerably smaller number of optimization variables, so the number of possible damage locations can be easily increased, and this is the most important benefit of this approach (see Ravaii et al., 1998b).

To demonstrate the potential of this approach, this method has been applied to damage detection in the frame for the mild type of damage which results in small changes in natural frequencies. The number of possible damage points was assumed to be first 15 and, in the second attempt, 31 and, accordingly, the upper bounds of the variables x_1^j were taken as 16 and 32, respectively. The lower bounds of these variables were taken as 1. The lower and upper bounds of cross-

sectional areas, x_2^j, were taken as 1 and 128 respectively. The assumed maximum number of damaged elements, L, was assumed to be one, two and three in three successive damage detection trials. In all cases the damage was successfully detected, the results are shown in Tables 5 and 6.

Table 5. Damage detection using GA (15 possible locations)

		Damage location and extent						
Element No.		1	2	3	4	5	6	7-15
Exact area (%)		100	64	100	100	100	100	100
Assumed max. number of damage locations *(L)*	*L*=1	100	67	100	100	100	100	100
	L=2	100	67	100	100	100	100	100
	L=3	100	64	98	100	103	100	100

Table 6. Damage detection using GA (31 possible damage locations)

		Damage location and extent									
Element No.		1	2	3	4-8	9	10	11	12-14	15	16-31
Exact area (%)		100	64	100	100	100	100	100	100	100	100
Assumed max. no. of damage loc. *(L)*	*L=1*	100	67	100	100	100	100	100	100	100	100
	L=2	58	100	100	100	103	100	100	100	100	100
	L=3	63	100	100	100	100	100	104	100	97	100

4 The Use of Permutation GA for the Development of Uniform Designs of Experiments

4.1 Introduction

Response surface modeling (see Myers and Montgomery, 1976), also referred to as metamodelling, is a method for approximating system's responses using function values at certain points in the design variable space. It is often used in design optimization for two main reasons: (i) to minimize the number of response evaluations, and (ii) to reduce the effect of numerical noise.

The choice of location of the evaluation points or plan points is important in getting a good approximation of the response, especially when evaluations are expensive. The methodologies used for formulating the plan points are collectively known as Design of Experiments (DOE). The most popular methods described by Myers and Montgomery (1976) and Box and Draper (1987) are mostly based upon the features of the mathematical model of the process, e.g. polynomial type. Another method is the Latin Hypercube sampling method (LH), proposed by McKay et al. (1979) and Iman and Conover (1980), which is independent of the mathematical model of a problem.

The LH DOE is structured so that each variable is divided into P levels. For each level, there is only one point (i.e. experiment). Once the DoE for N variables and P levels is formulated, re-calculation of the DOE is not required. Figure 10 shows the DoE for N=3 and P=4. The matrix is scaled to fit any range of the design variables. Therefore a LH DOE for a problem with N=3 and P=4 is generally determined by this matrix.

Point	x_1	x_2	x_3
1	1	1	4
2	2	3	1
3	4	4	2
4	3	2	3

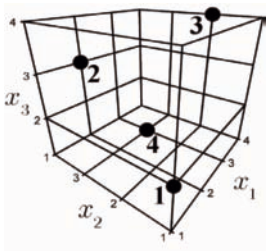


Figure 10. LH DoE for N=3 and P=4

Two LH methods are the random sampling LH method (RLH) and the optimal Latin Hypercube designs (OLH). RLH and OLH differ by how the points in the DOE are distributed. The RLH method uses random sampling to get each point in the DOE, whereas the OLH methods use more structured approaches with the aim

of optimizing the uniformity of the distribution of the points. The generation of the OLH DOE by enumeration is infeasible due to a very large number of possible combinations and therefore solving this minimization problem requires a more advanced optimization technique to search the design space.

4.2 Optimal Latin hypercube design of experiments

Several methods have been proposed to generate OLH using criteria such as maximizing entropy (Shewry and Wynn, 1987), integrated mean-squared error (Sacks et al., 1989), and the maximization of the minimum distance between points (Johnson et al., 1990). Jin et al. (2003) introduce an enhanced stochastic evolutionary algorithm for formulating OLH. Audze and Eglais (1977) proposed a method (abbreviated here as AELH) that uses the potential energy of the points in the DOE to generate a uniform distribution of points. The AELH objective function is used in this section.

Audze-Eglais objective function

The Audze-Eglais method is based on the following physical analogy: a system consisting of points of unit mass exert repulsive forces on each other so that the system has a certain amount of potential energy. When the points are released from an initial state, they move. They will reach equilibrium when the potential energy of the repulsive forces between the masses is at a minimum. If the magnitude of the repulsive forces is inversely proportional to the distance squared between the points then minimizing the function

$$\sum_{p=1}^{P} \sum_{q=p+1}^{P} \frac{1}{L_{pq}^{2}} \rightarrow \min \tag{6}$$

will produce a system of points distributed as uniformly as possible. Here U is the potential energy and L_{pq} is the distance between the points p and q ($p \neq q$). For two design variables (N=2) and three points (P=3) the design of experiments shown in Figure 11 is one possible solution to the OLH DOE. The quality of the solution is calculated using the objective function in (6).

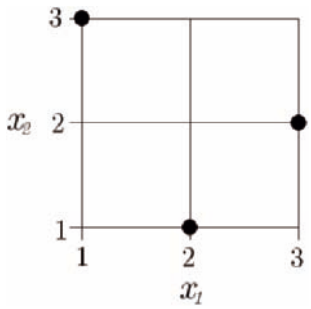


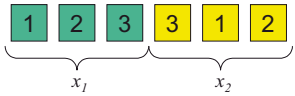
Figure 11. DOE for N=2 and P=3

Various DOE combinations can be evaluated and the DOE that minimizes the objective function is the OLH DOE. As the problem is discrete, it is ideally suited to the use of discrete optimization techniques such as genetic algorithms.

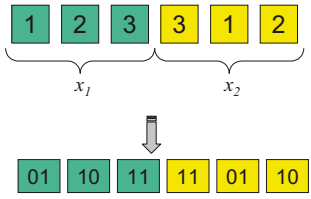
Encoding for use in a genetic algorithm

Optimization requires the design variables of the problem to be encoded. The design variables for this problem are the points of the DOE, so encoding of these points into a form understood by the optimizer is required.

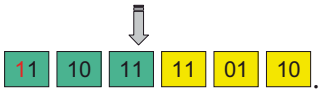
The approach used here encodes the co-ordinates of each point. The first P design variables being the x_1 co-ordinates, the next P design variables being the x_2 co-ordinates etc. up to N variables. So the encoding or 'permuation' of Figure 11, i.e. (1,3) (2,1) (3,2) becomes



The use of a standard binary genetic algorithm requires the encoding to be binary, e.g.



if a standard genetic operator is applied, such as a mutation to the first point, this becomes:



It can be seen that the mutation results in a contravention of the LH rule of one point in each level, therefore penalization of the solution is required to guide the binary GA towards feasible designs. A penalization approach is highly undesirable for a GA and is very inefficient (Bates et al. 2004). One approach to ensure that no members of a population need to be penalized is to use a permutation GA (permGA), see Michalewicz (1992). This approach has been introduced by Bates et al. (2004) and Toropov et al. (2007) and is described here together with some further recent developments and extensions. This section is split into three subsections corresponding to the following developments:

- A method using a permutation genetic algorithm (Michalewicz, 1992) for generating a standard OLH (permGA) is presented and is adapted from Bates et al. (2004).
- This is then extended to allow for existing fixed points to be incorporated into the OLH.
- A strategy is described for the treatment of problems where different design variables are associated with different numbers of levels which is not covered by the standard definition of a Latin hypercube design.

4.3 Optimal Latin Hypercube Generation Using a Permutation Genetic Algorithm

The requirement of one point in each level for OLH is similar to the travelling salesman problem (TSP). The only difference between the TSP and the formulation of the LH is that in the LH problem the 'salesman' doesn't return to the starting point. An extensive overview of many approaches to solving the TSP problem is given by Michalewicz (1992). One method is to use a GA to find the optimal 'permutation' of cities where the term permutation, is "the rearrangement of existent elements, or the process of changing the lineal order of an ordered set of objects" (Merriam-Webster online dictionary).

Using permGA the encoding is done with integer values instead of binary numbers. Furthermore, the mutation and crossover operators are modelled such that the rule of one point in each level for the LH is never contravened. This means that the optimization problem is unconstrained and no penalty factor is required. Therefore, using a method such as permGA is more efficient due to the fact that it does not have to deal with infeasible solutions.

Encoding

Using the permGA the encoding requires no conversion so that the string representing the DOE in Figure 12 is 1 2 3 3 1 2. The formulation here is that the first P numbers are a random sequence of numbers between 1 and P, and the next P numbers are also a random sequence of numbers between 1 and P. This is repeated up to N times. There are no repetitions of numbers in each sequence therefore the rule of one point in each level is not contravened. Below is an example of using the genetic operators with a permutation encoding.

1) Mutation - two numbers are selected and exchanged, e.g. 2nd and 5th

$$[4\ \mathbf{1}\ 5\ 2\ \mathbf{3}] \quad \Rightarrow \quad [4\ \mathbf{3}\ 5\ 2\ \mathbf{1}]$$

2) Crossover can be done in a variety of ways. It is applied to each sequence of P numbers for the N variables. Three crossover methods have been implemented in this work: a 'simple crossover' method, the 'cycle crossover' method by Oliver et al. (1987), and an 'inversion' method.

Simple crossover

A crossover point is selected (2 in this example), the permutation is copied from the first parent until the crossover point, then, starting from the beginning, the other parent is scanned and if the number is not yet in the child, it is added.

$$\begin{array}{lcl} \text{Parent 1} = [\mathbf{4\ 1}\ 3\ 5\ 2] & \Rightarrow & \text{Child 1} = [\mathbf{4\ 1}\ 5\ 2\ 3] \\ + & & \\ \text{Parent 2} = [\mathbf{5\ 2}\ 1\ 4\ 3] & \Rightarrow & \text{Child 2} = [\mathbf{5\ 2}\ 4\ 1\ 3] \end{array}$$

Cycle crossover

In this method, each value and its position comes from one of the parents. The method preserves the absolute position of the elements in the parent sequence. An example of the implementation of cycle crossover is as follows (adapted from Michalewicz, 1992).

$$\begin{array}{c} \text{Parent 1} = [1\ 3\ 9\ 7\ 5\ 4\ 6\ 2\ 8] \\ + \\ \text{Parent 2} = [4\ 6\ 2\ 1\ 7\ 8\ 9\ 3\ 5] \end{array}$$

produces child 1 by taking the first value from the first parent:

$$\text{Child 1} = [1\ *\ *\ *\ *\ *\ *\ *\ *].$$

The next point is the value below 1 in parent 2, i.e. 4. In parent 1 this value is at position '6', thus

$$\text{Child 1} = [1 * * * * 4 * * *].$$

This, in turn, implies a value of 8, as the value from parent 2 below the selected value 4. Thus

$$\text{Child 1} = [1 * * * * 4 * * 8].$$

Following this rule, the next values in child 1 are 5 and 7. The selection of 7 requires the selection of 1 in parent 1, which is already used, meaning that a cycle is completed

$$\text{Child 1} = [1 * * 7 5 4 * * 8].$$

The remaining values are filled from the other parent:

$$\text{Child 1} = [1 6 2 7 5 4 9 3 8].$$

Similarly,

$$\text{Child 2} = [4 3 9 1 7 8 6 2 5].$$

Inversion

In this method two cut-off points are chosen at random in a parent and the values between these points are inverted e.g. for the cut-off points 3 and 7 marked as '|'

$$\text{Parent 1} = [1\ 3\ 9 \mid 7\ 5\ 4\ 6 \mid 2\ 8]$$
$$\Downarrow$$
$$\text{Child 1} = [1\ 3\ 9 \mid 6\ 4\ 5\ 7 \mid 2\ 8]$$

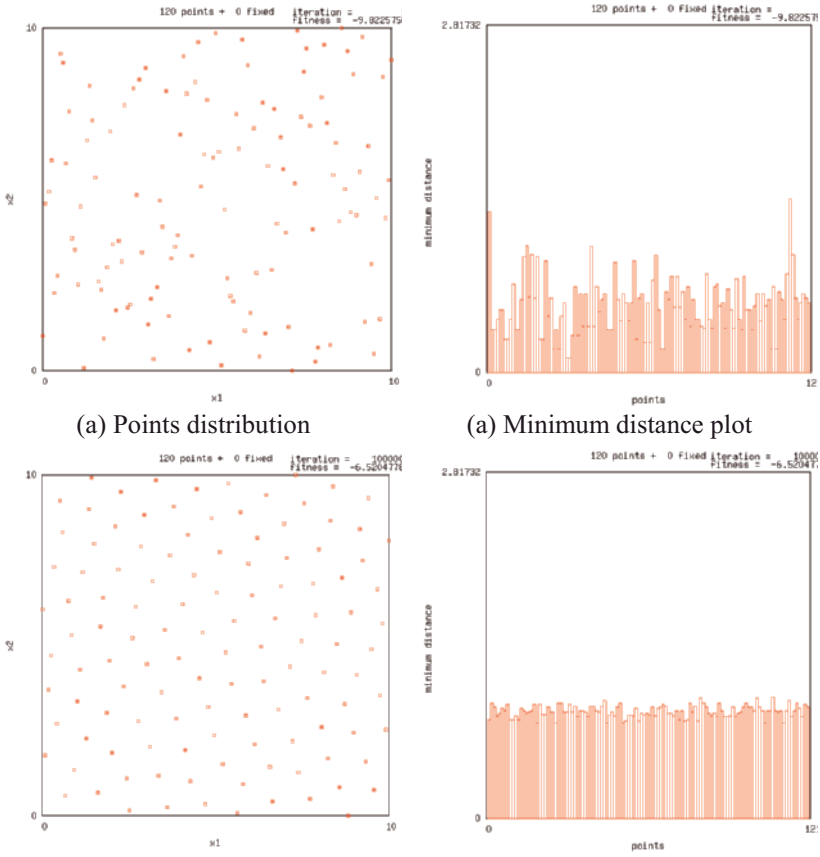
.

Other crossover methods include 'partially mapped crossover' by Goldberg and Lingle (1985) and 'order crossover' by Davis (1985), also see Michalewicz (1992) for further details.

In addition to using the objective function (6) to assess the fitness of a solution it is useful to find for each sample point in a DOE the location of its nearest neighbour by calculating the minimum Euclidian distance out of all DOE points. Plotting the distribution of the "minimum distances", a smaller standard devia-

tion demonstrates improved uniformity and a higher mean indicates improved space filling propertiy.

Figure 12 compares a random LH to an OLH for 2 design variables and 120 points generated using permGA.



(a) Points distribution (a) Minimum distance plot

(b) Points distribution (b) Minimum distance plot

Figure 12. Comparison of (a) random LH to (b) OLH for 2 design variables and 120 points generated using permGA

4.4 Optimal Latin hypercube design with existing fixed points

One of the shortcomings of LH DOEs is that it is not possible to have points at all the extremities (corner points) of the design space due to the rule of one point

per level. This is often desirable in practical design optimization problems. Furthermore, it may be the case that points in the design space are pre-designated. With this in mind the standard OLH procedure described in Section 4.2 has been extended to allow existing fixed points to form part of the final DOE. This has been implemented by allowing the fixed points to contribute to the objective function of the optimization without forming part of the design variable set. This enables fixed points to be located anywhere in the design domain without affecting the fundamental principle of the designable Latin hypercube i.e. one point per level per design variable.

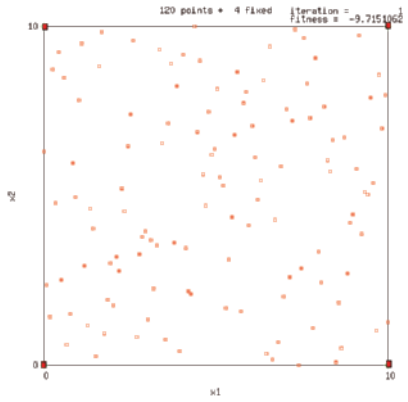
Figure 13 compares a random LH to an OLH for 2 design variables, 4 fixed (corner) points and 120 points generated by the permGA. Similarly, Figure 14 compares a random LH to a OLH for 2 design variables, 11 fixed (diagonal) points and 120 points generated by the permGA.

4.5 Optimal Latin hypercube design with different numbers of levels

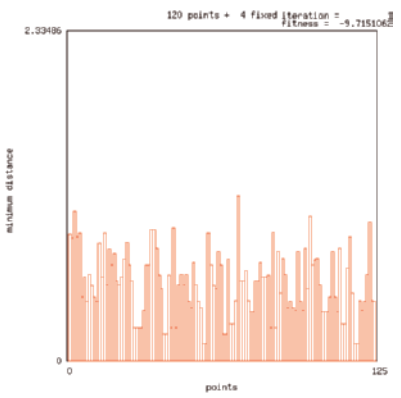
Another shortcoming of LH DOEs is that it is not possible to have different numbers of levels in different design variables. In many practical problems design variables are defined on discrete sets which do not necessarily contain the same number of possible values for each of the individual variables. In such a case, a conventional Latin hypercube is not applicable as some levels in some of the design variables would have more than one point. With this in mind the OLH procedure described in Section 1 has been extended to allow individual levels to contain more than one point while preserving the uniformity property according to the Audze-Eglais optimality criterion.

In order to do this within the adopted framework, a set of possible levels in each of design variables has to be defined. The total number of such possibilities has to be the same for each of the design variables but some of the levels can be repeated more than once. For example, for a two design variable problem with 9 points, 9 levels in the first variable and 3 levels in the second, the sets are: {1, 2, 3, 4, 5, 6, 7 8, 9} in the variable one and {1, 1, 1, 2, 2, 2, 3, 3, 3} in the variable two. Then the procedure described in Section 4.3 can be directly applied for such a problem. Figure 15 shows a generalised OLH design obtained by a permGA.

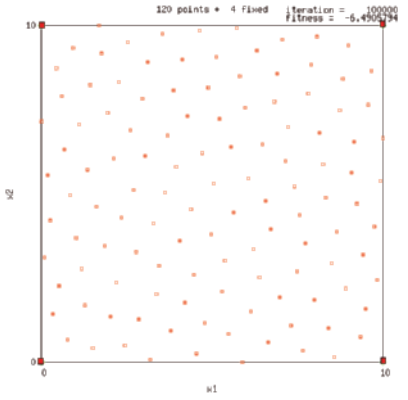
Another example is a generalised OLH with two design variables, 30 points, 30 levels in the first variable and 20 levels in the second. The set of levels for the first design variable includes integers from 1 to 30, and for second it was chosen as {1, 2, 3, 4, 5, 6, 7 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2, 4, 6, 8, 10,



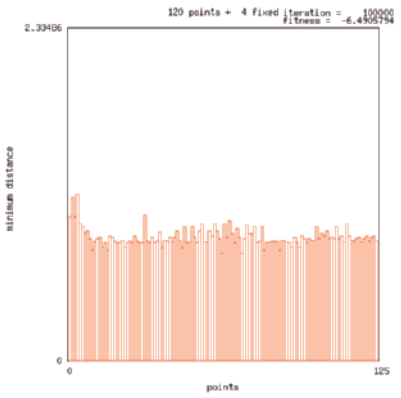
(a) Points distribution



(a) Minimum distance plot



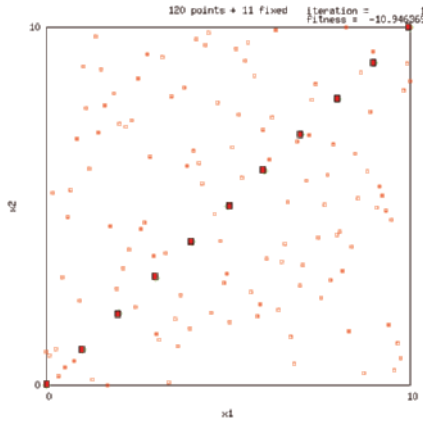
(b) Points distribution



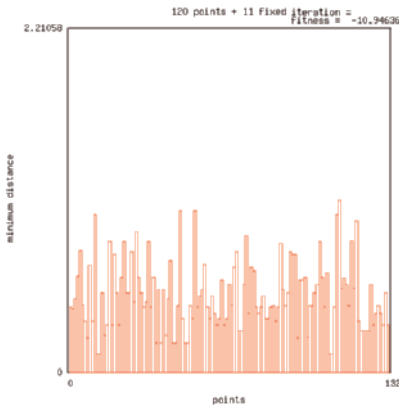
(b) Minimum distance plot

Figure 13. Comparison of (a) random LH to (b) generalised OLH for 2 design variables, 4 fixed (corner) points and 120 points generated by the permGA

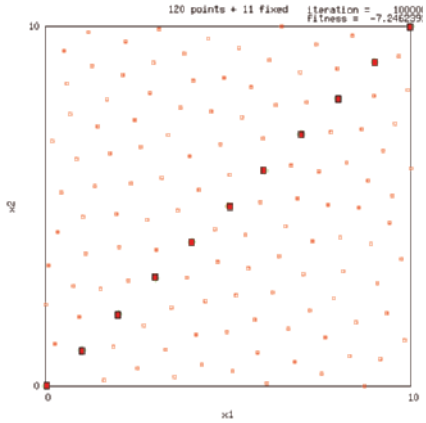
12, 14, 16, 18, 20}. Figure 16 shows a corresponding generalised OLH design obtained by a permGA. These example indicate how a sets of levels is established for each of the design variables: the full set of available levels is repeated as many times as possible not exceeding the given number of DOE points, and then the remaining levels are chosen by picking up a necessary number of levels (the total number of all levels is to remain equal to the number of DOE points) out of the available set covering it as uniformly as possible.



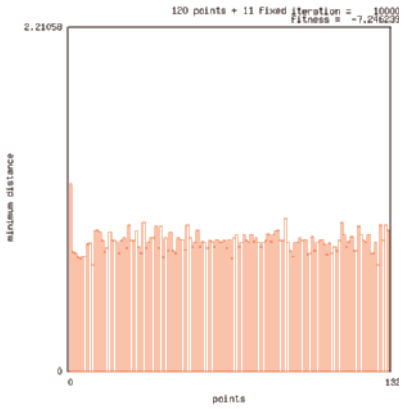
(a) points distribution



(a) Minimum distance plot



(b) points distribution



(b) Minimum distance plot

Figure 14. Comparison of (a) random LH to (b) generalised OLH for 2 design variables, 11 fixed (diagonal) points and 120 points generated by the permGA

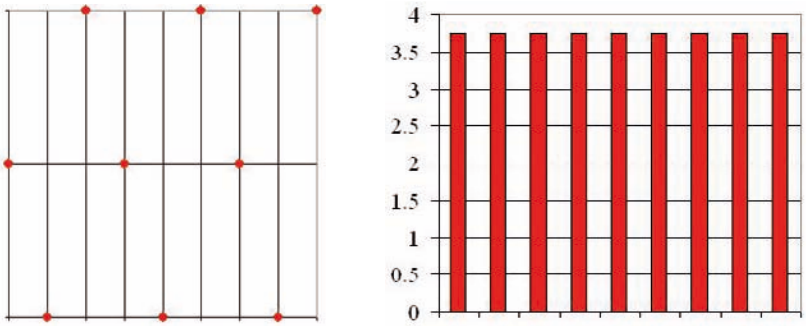


Figure 15. Generalised OLH for 2 design variables, 9 levels in the first and 3 levels in the second design variable

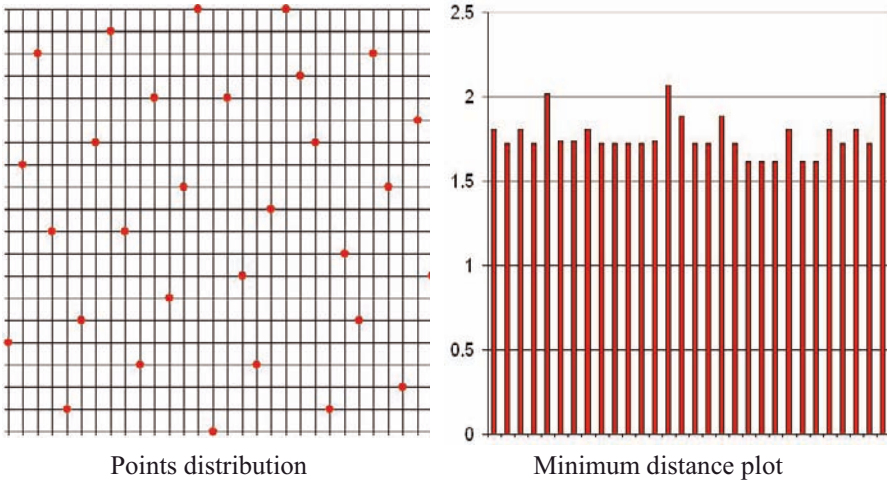


Figure 16. Generalised OLH for 2 design variables, 30 levels in the first and 20 levels in the second design variable

4.6 Conclusions

A method has been developed for formulating the OLH DOE using the Audze-Eglais objective function. The formulation of this DOE is shown to be non-trivial. It has been shown, that the formulation of OLHs is ideally suited to using

permutation GAs since the problem uses discrete design variables and the LH requires no repetition of values in a chromosome.

One of the shortcomings of LH DOEs is that it is not possible to have points at all the extremities (corner points) of the design space due to the rule of one point per level. Furthermore, it may be the case that points in the design space are pre-designated. The permutation GA developed has been extended to account for such situations. Next, a strategy is developed for the treatment of problems where different design variables are associated with different numbers of levels. This addresses a limitation of the standard convention of a Latin hypercube design whilst preserving the uniformity property.

Overall, it can be concluded, that the permutation GA is an effective tool for developing OLH DOE and that the extensions described considerably increase the functionality of the tool.

5 Use of Genetic Programming Methodology for Metamodel Building

Selection of the structure of an analytical metamodel, i.e. an approximation function, is a problem of empirical model building (Box and Draper, 1987). Selection of individual regression components in a model results in solving a combinatorial optimization problem. Even if the bank of all regressors is established (which is a difficult problem on its own), the search through all possible combinations would result in prohibitive computational effort. Toropov and Alvarez (1998) attempted to develop and use a genetic programming (GP) methodology (Koza, 1992, Kinnear, 1994) for the creation of an approximation function structure of the best possible quality, and use it within a mid-range (or global) approximation technique.

GP is a branch of genetic algorithms (GA). While a GA uses a string of numbers to represent the solution, the GP creates a population of computer programs with a tree structure. In our case of design optimization, a program represents an empirical model to be used for approximation of a response function. A typical program, representing the expression $(x_1/x_2+x_3)^2$, is shown in Figure 17.

These randomly generated programs are general and hierarchical, varying in size and shape. GP's main goal is to solve a problem by searching highly fit computer programs in the space of all possible programs that solve the problem. This aspect is the key to finding near global solutions by keeping many solutions that may potentially be close to minima (local or global). The creation of the initial population is a blind random search of the space defined by the problem. In contrast to a GA, the output of the GP is a program (i.e. an empirical model used for approximation), whereas the output of a GA is a quantity.

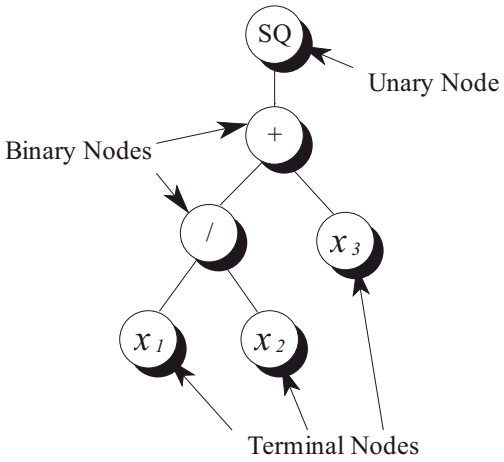


Figure 17. Typical tree structure for $\left(\frac{x_1}{x_2}+x_3\right)^2$

The programs are composed of elements from a *terminal set* and a *functional set*, called nodes.

Terminal Set	Design variables: x_1 , x_2 , ..., x_N
Functional Set	Mathematical operators that generate the regression model: { +, *, /, x^y, etc. }

The functional set can be subdivided into *binary nodes*, which take any two arguments (like addition), and *unary nodes*, which take one argument, e.g. a square root. All the functions and terminals must be compatible in order to faultlessly pass information between each other (closure property).

The evolution of the programs is performed through the action of the genetic operators and the evaluation of the fitness function.

5.1 Genetic operators

Model structures evolve through the action of three basic genetic operators: reproduction, crossover and mutation. In the *reproduction* stage, a strategy must be adopted as to which programs should die. In this implementation, trees with

fitness below the average are killed. The population is then filled with the surviving trees according to fitness proportionate selection.

Crossover (Figure 18) combines good information from two trees (parents) in order to improve the fitness of the next generation. The basic algorithm is as follows:

- select two trees from the whole population;
- within each of these trees, randomly select one node;
- swap the subtrees under the selected nodes, thus generating two offspring belonging to the new population.

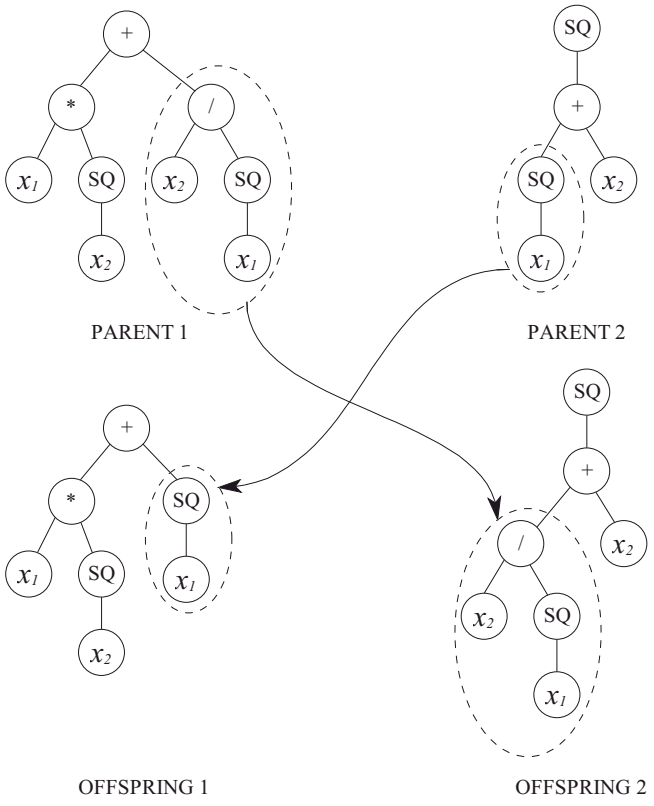


Figure 18. Crossover

Mutation (Figure 19) protects the model against premature convergence and improves the non-local properties of the search. The following algorithm is used:

- randomly select one node within a tree;
- replace this node with another one from the same set (a function replaces a function and a terminal replaces a terminal) except by itself.

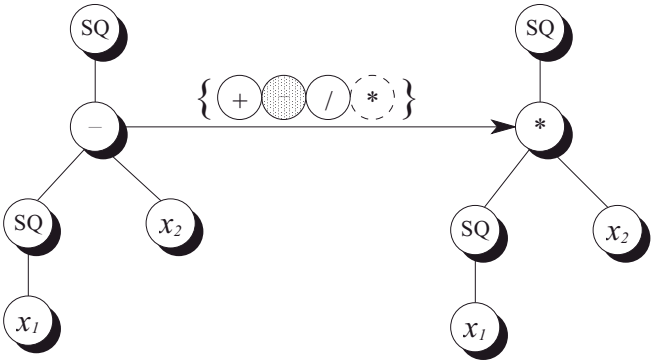


Figure 19. Mutation

An additional operator, *elite transfer*, is used to allow a relatively small number of the fittest programs, called the elite, to be transferred unchanged to a next generation, in order to keep the best solutions found so far. As a result, a new population of trees of the same size as the original one is created, but it has a higher average fitness value.

5.2 Fitness function

When selecting randomly a tree to perform any genetic operation, the so-called fitness proportionate method is used here. This method specifies the probability of selection on the basis of the fitness of the solution.

The fitness of a solution shall reflect (*i*) the quality of approximation of the experimental data by a current expression represented by a tree, and (*ii*) the length of the tree in order to obtain more compact expressions.

In problems of empirical model building, the most obvious choice for the estimation of the quality of the model is the sum of squares of the difference between the analytical model output (that is being built) and the results of runs of the original simulation model (or experimental information) over some chosen

design of experiments (DOE). In a dimensionless form this measure of quality of the solution can be presented as follows:

$$Q(S_i) = \frac{\sum_{p=1}^{P} \left(F_p - \widetilde{F}_p\right)^2}{\sum_{p=1}^{P} F_p{}^2}. \tag{7}$$

If in addition to the values of the original function F_p their first order derivatives at point p $F_{p,i} = \frac{\partial}{\partial x_i} F_p$, $(i = 1,...,N)(p = 1,...,P)$ are known, the numerator in (7) is replaced by the following expression:

$$\sum_{p=1}^{P} \left(\left(F_p - \widetilde{F}_p\right)^2 + \gamma \frac{\sum_{i=1}^{N} \left(F_{p,i} - \widetilde{F}_{p,i}\right)^2}{\sum_{i=1}^{N} F_{p,i}{}^2} \right) \tag{8}$$

where γ>0 is the parameter characterizing a degree of inequality of the contribution of the response and the sensitivity data, taken here as 0.5.

If $Q(S_i)$ is the measure of quality of the solution S_i, $Q_{\max}$ is the maximum value of this quantity out of all N_t members of the population, $ntp_{\max}$ is the maximum allowed number of tuning parameters, ntp_i is the number of tuning parameters contained in the solution S_i and c is a coefficient penalizing the excessive length of the expression, the fitness function $\Phi(S_i)$ can be expressed in the following form:

$$\Phi(S_i) = Q_{\max} - Q(S_i) + c * \left(ntp_{\max} - ntp_i\right)^2 \tag{9}$$

The probability that the solution S_i will be selected is

$$\frac{\Phi(S_i)}{\sum_{j=1}^{N} \Phi(S_j)}. \tag{10}$$

Programs with greater fitness values $\Phi(S_i)$ have a greater chance of being selected in a subsequent genetic action. Highly fit programs live and reproduce, and less fit programs die.

The fitness function defined as the sum of squares of the error between model output and experimental data has been applied before to the model building in fluid flow problems (Gray et al., 1996a) and dynamic systems (Gray et al., 1996b). Another possible choice is the statistical concept of correlation, which determines and quantifies whether a relationship between two data sets exists. This definition was used for the identification of industrial processes (McKay et al. 1996).

5.3 Design of experiments

The choice of the design of experiments can have a large influence on the accuracy of the approximation and the cost of constructing the response surface. In this work, the approach suggested by Audze and Eglais (1977), see Section 4 for details.

5.4 Model tuning

The approximation function is characterized not only by its structure (to be found by the GP) but also by a set of tuning parameters $\boldsymbol{a}$ to be found by the model tuning, i.e. the least squares fitting of the model into the set of values of the original response function:

$$G(\boldsymbol{a}) = \sum_{p=1}^{P} \left(F_p - \tilde{F}_p(\boldsymbol{a})\right)^2 \rightarrow \min \tag{11}$$

The allocation of tuning parameters $\boldsymbol{a}$ to an individual tree (Figure 20) follows the basic algebraic rules, see Figure 21.

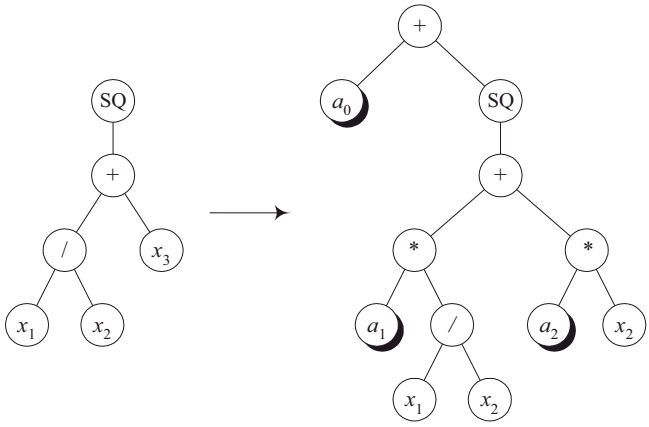


Figure 20. Allocation of tuning parameters to a tree

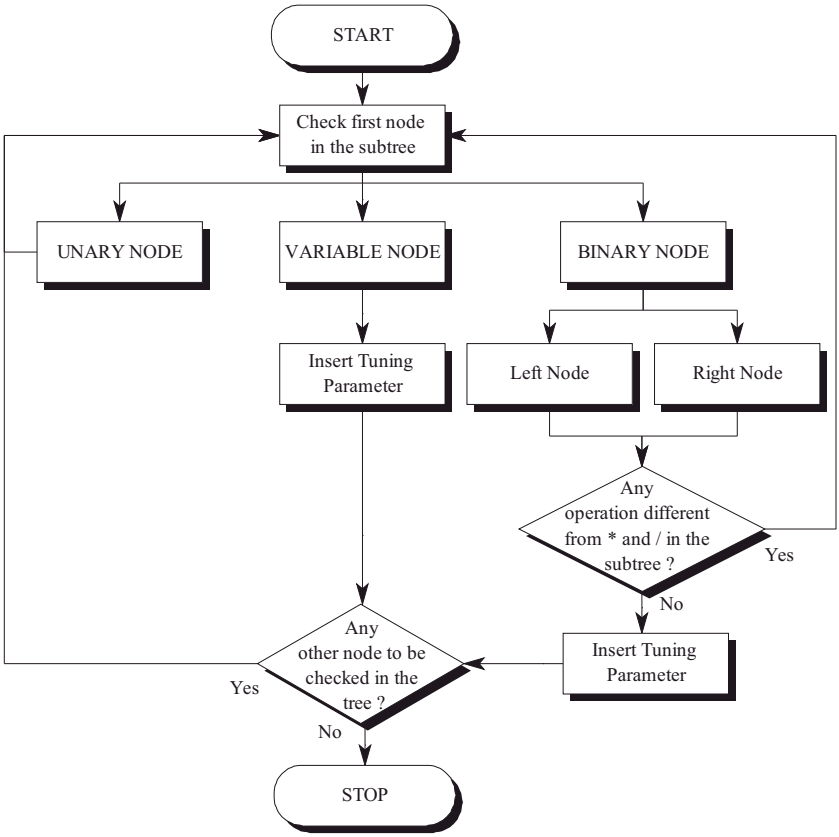


Figure 21. Tuning parameter allocation diagram

Going through the tree downwards, tuning parameters are allocated to a subtree depending on the type of the current node and the structure of the subtree according to the following algorithm:

1. Current node is of type *Binary*

– multiplication and division operations only require one tuning parameter, e.g.

$$\widetilde{F} = x_1 * x_2 \quad \Rightarrow \quad \widetilde{F}(\boldsymbol{a}) = a_1 * x_1 * x_2$$

– all other operations require two tuning parameters, e.g.

$$\widetilde{F} = x_1 + x_2 \quad \Rightarrow \quad \widetilde{F}(\boldsymbol{a}) = a_1 * x_1 + a_2 * x_2$$
$$\widetilde{F} = x_1 \wedge x_2 \quad \Rightarrow \quad \widetilde{F}(\boldsymbol{a}) = (a_1 * x_1) \wedge (a_2 * x_2)$$

– all other operations require two tuning parameters, e.g.

$$\widetilde{F} = x_1 + x_2 \quad \Rightarrow \quad \widetilde{F}(\boldsymbol{a}) = a_1 * x_1 + a_2 * x_2$$
$$\widetilde{F} = x_1 \wedge x_2 \quad \Rightarrow \quad \widetilde{F}(\boldsymbol{a}) = (a_1 * x_1) \wedge (a_2 * x_2)$$

– all other operations require two tuning parameters, e.g.

$$\widetilde{F} = x_1 + x_2 \quad \Rightarrow \quad \widetilde{F}(\boldsymbol{a}) = a_1 * x_1 + a_2 * x_2$$
$$\widetilde{F} = x_1 \wedge x_2 \quad \Rightarrow \quad \widetilde{F}(\boldsymbol{a}) = (a_1 * x_1) \wedge (a_2 * x_2)$$

– when $\widetilde{F}$ is a combination of the previous two approaches, tuning parameters are only applied to operations different from multiplication and division, e.g.

$$\widetilde{F} = x_1 * (x_2 / x_3 + x_4) \quad \Rightarrow$$
$$\widetilde{F}(\boldsymbol{a}) = x_1 * (a_1 * x_2 / x_3 + a_2 * x_4)$$

$$\widetilde{F} = (x_1 + x_2) \wedge (x_3 * x_4) \quad \Rightarrow$$
$$\widetilde{F}(\boldsymbol{a}) = (a_1 * x_1 + a_2 * x_2) \wedge (x_3 * x_4)$$

2. Current node is of type *Unary*: ignore.
3. Current node is of type *Variable*
   – one tuning parameter is added, e.g.

$$\widetilde{F} = (x_1)^2 \quad \Rightarrow \quad \widetilde{F}(\boldsymbol{a}) = (a_1 * x_1)^2$$

4. Insert a free parameter, e.g.

$$\widetilde{F}(a) = a_1 * x_1 \quad \Rightarrow \quad \widetilde{F}(a) = a_1 * x_1 + a_0$$

To identify the parameters of the expression by the nonlinear least squares fitting, i.e. to solve the optimization problem (11), a combination of a Genetic Algorithm and a nonlinear optimization method by Madsen and Hegelund (1991) is used.

The output of the Genetic Algorithm is then used as the initial guess for the subsequent derivative-based optimization. The method by Madsen and Hegelund (1991) used here amounts to a variation of the Newton's method in which the Hessian matrix is approximated by the secant (quasi-Newton) updating method. Once the technique comes sufficiently close to a local solution, it normally converges quite rapidly. To speed up the convergence the algorithm uses the adaptive update of the Hessian and, consequently, the algorithm is reduced to either a Gauss-Newton or Levenberg-Marquardt method.

5.5 Applications

Test example 1. Rosenbrock's banana-shaped valley function is a classic optimization problem. The optimum point is inside a long, curved and narrow valley. The function is defined as follows:

$$F(x_1, x_2) = 100 * \left(x_2 - {x_1}^2\right)^2 + \left(1 - x_1\right)^2$$

Figure 22 shows the contour plot and the surface plot of Rosenbrock's function.

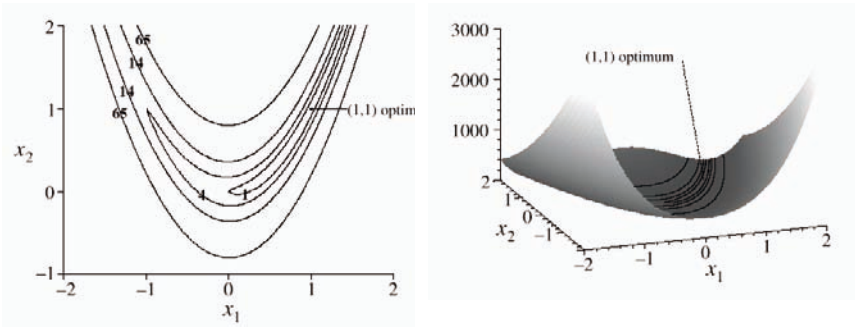


Figure 22. Rosenbrock's function: contour plot (left) and surface plot (right)

With a population of 200 trees, the approximation of Rosenbrock's function has been tested with and without the use of sensitivity information. When no sensitivity information has been used, GP was run with a DOE of 5 and 10

points. In the case of 5 points, a solution with good fitness has evolved, it had almost exact match at the plan points but very poor quality everywhere else. The reason is that insufficient information was passed to GP to represent an accurate solution, and the solution suffered from overfitting. When approximated with 10 plan points, the Rosenbrock's function emerged as the solution of the problem.

When the first order derivatives, were included in the approximation of Rosenbrock's function with a DOE of 5 points, the algorithm exactly matched the original expression. This suggests that, if available, derivatives provide with more information thus improving the convergence characteristics. If the derivatives are not available, the inclusion of more points in the plan of experiments is necessary.

Test example 2. Generally, a large number of DOE points is desirable in order to provide more information to the genetic programming algorithm. To illustrate these aspects, the following expression has been tested (see Figure 23):

$$(30 + x_1 \sin x_1)(4 + e^{-x_2})$$

Two tests were performed with data generated with a DOE of 20 and 10 points (Figures 23 and 24 respectively). The sine and exponential functions were included in the functional set. Results show that the higher is the number of experiments, the better is the approximation.

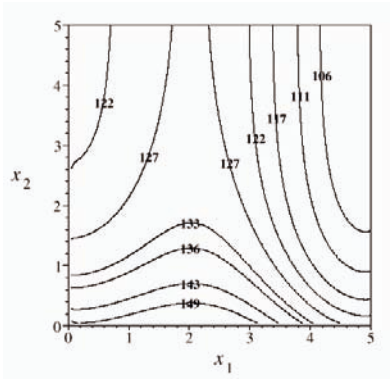


Figure 23. The original function and the approximation with 20 DOE

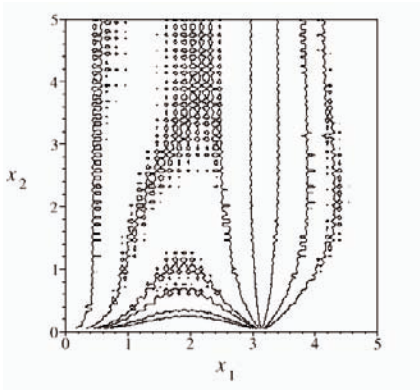


Figure 24. Approximation with 10 point DOE

6 Use of Genetic Programming for Recognition of Damage in Steel Structures

In this section the output error method of system identification has been again used to assess the presence and extent of damage in steel structures. In the solution of the optimization problem the objective function and, alternatively, its individual terms corresponding to individual frequencies, have been approximated by analytical expressions using the Genetic Programming methodology. Damage location in a typical steel portal frame is found by minimization of the difference between the measured and analysed structural response, namely, frequencies of vibration (5).

In the formulation of the optimization problem (5) the number of optimization variables $N = 3$, the number of used frequencies $M = 4$, and x_1, x_2, x_3 describe percentage of reduction of cross-sectional area in three locations at welded joints. The description of actual damage corresponds to the following set of optimization variables: $x_1 = 100$, $x_2 = 54$, $x_3 = 100$, i.e. damage in second location.

The approximation procedure using GP (see Section 5) has been carried out following two different approaches (Toropov et al. 1999 a, b): approximation of the objective function in the original optimization problem (5), and approximation of the individual frequencies corresponding to the first four modes of vibration. In the second approach, individual frequencies $\omega_i^a(\boldsymbol{x})$, $i = 1, \ldots, M$ in (5) are approximated by simpler expressions $\widetilde{\omega}_i^a(\boldsymbol{x})$ and the overall objective function (to be minimized) $\widetilde{F}(\boldsymbol{x})$ can be assembled similarly to (5) using the approximated frequencies:

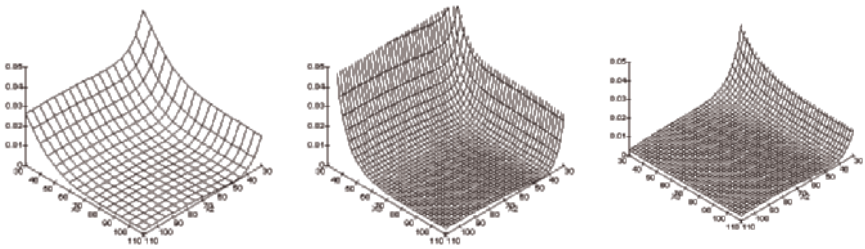
$$\widetilde{F}(\boldsymbol{x}) = \sum_{i=1}^{M} w_i \left\{ \left[\omega_i^m - \widetilde{\omega}_i^a(\boldsymbol{x}) \right] / \omega_i^m \right\}^2 \rightarrow \min . \qquad (12)$$

The advantage of the formulation (12) is that the approximations $\widetilde{\omega}_i^a(\boldsymbol{x})$ can be built once and then used many times for damage detection in a new structure of the same geometry using new sets of the experimental data ω_i^m, $i = 1, \ldots, M$.

For the 3-dimensional graphical representation, the approximation functions have been plotted fixing one of the three optimization variables, corresponding to possible damage locations, i.e. $x_1 = 100$. Figure 26a shows the original function in (5). Figures 26b and 26c show the approximation functions obtained using the values of the function in (5) at $P = 20$ points of the optimization variable space. The following input parameters have been used:

- designs of experiments: 20 and 50 points
- population size: N_t=100
- proportion of the elite: P_e=0.2
- probability of mutation: P_m=0.001
- functional set:
- binary functions +, *, /, ^

– unary functions $(...)^2$, $\sqrt{(...)}$, $-(...)$
– terminal set: variables x_1, x_2, x_3.

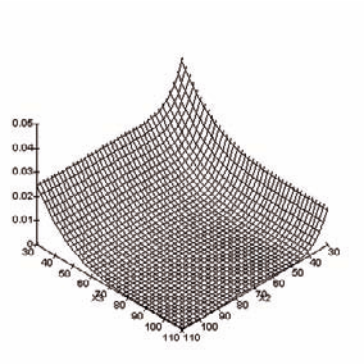


(a) Original function (b) Approximation (50 points) (c) Approximation (20 points)

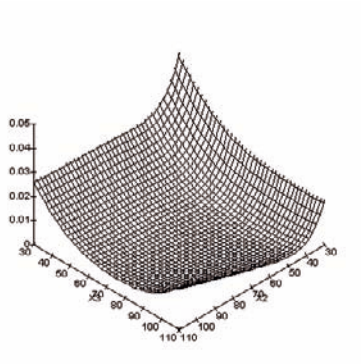
Figure 25. Approximation of the overall expression, $x_1 = 100$

The solution of the simplified optimization problem (5) has been obtained in two steps of approximation building. In the first step the following values of lower and upper bounds have been selected: $A_j = 10$ and $B_j = 110$, $j = 1, 2, 3$. In the second step the size of the search domain of the optimization variable space, defined by A_j and B_j, has been reduced by half and the new approximations have been constructed. When the approximation have been built using 50 points, the following solution has been obtained: $x_1 = 110.0$, $x_2 = 45.4$, $x_3 = 110.0$. Using 20 points, the following solution has been obtained: $x_1 = 74.0$, $x_2 = 50.7$, $x_3 = 110.0$.

When the approximation functions were obtained as a combination of approximations for the individual frequencies, as defined by the expression (12) and illustrated by Figure 26, the following solutions have been obtained in one step: $x_1 = 92.6$, $x_2 = 50.1$, $x_3 = 110.0$ using 50 points, and $x_1 = 80.0$, $x_2 = 51.1$, $x_3 = 89.6$ using 20 points.



Approximation (50 points)



Approximation (20 points)

Figure 26. Expressions obtained using approximations of individual frequencies, $x_1 = 100$

7 Multicriteria Optimization of the Manufacturing Process for Roman Cement using Genetic Programming

In the 19th century, Roman cement was used throughout Europe for the production of stucco finishes in architectural decoration. However, problems with the supply of raw materials and developments in Portland cements motivated the decline in its use and the corresponding loss of craft techniques.

The Charter of Venice (1964) states that the process of restoration should be based on respect for the original material. This is in contrast with the use of the current modern cement products, finishing layers and paints that do not match the original physical and mechanical properties of the stuccoes.

Consequently, for the re-introduction of Roman cement, there is a need to find a suitable range of similar materials among the re-emerging natural hydraulic binders, and to appreciate the technical knowledge and understanding of the original cement-makers. Experimental results on the calcination of cement-stones from the Harwich and Whitby group of cements show that both setting time and strength development are functions of source and calcination temperature.

In this application, a single source of cement-stone was identified for experimentation within an optimization programme to relate mechanical and mineralogical characteristics to calcination conditions. Genetic programming has been used to illustrate the general trends of minerals and the strength development of the cement. The data will be useful for the selection of hydraulic binders and as an element in the re-introduction of Roman cement to the European market.

Experimental work. The cement-stones used in this research were collected from the Yorkshire coast at Whitby Long Bight. The calcination process is not well documented in the historic literature. For the current research, the Audze-Eglais 12 point DOE has been used. Each cement is referred to using the nomenclature of temperature and residence time, e.g. 917/276 taken as design variables x_1 and x_2 respectively, see Figure 27. An electric kiln was used and no attempt made to either circulate air through it or to seal it during calcination.

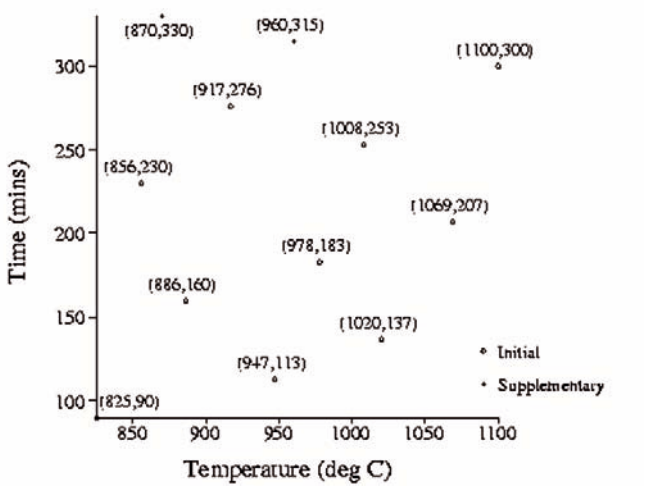


Figure 27. Design of experiments

Results presented here are adopted from Alvarez et al. (2000). At each combination of time and temperature given by the plan of experiments, strength at one week (*MPa*) and rate of strength enhancement between 8 and 17 weeks (MPa/week) have been obtained by a series of laboratory experiments. Using GP, these response quantities have been obtained as analytical expressions, their contour plots are shown in Figure 28.

The multicriteria optimization problem is stated as to maximize the early age strength and the rate of strength enhancement simultaneously. The constraints have been established following different technical aspects. Although the green discolouration of some pastes is associated with the presence of a mineral called calcium sulfite, its concentration is too low to quantify. Consequently this constraint has been represented by a maximum silica content (determined by X-Ray Difraction using boehmite as an internal standard) expressed as relative intensity

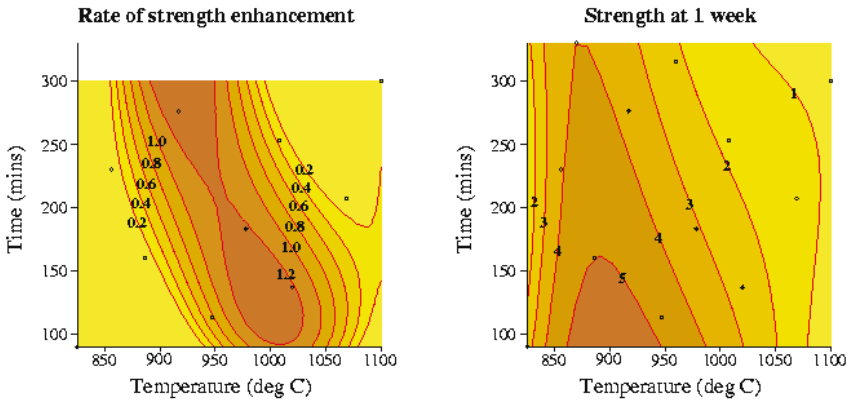


Figure 28. Strength development for Roman cement

50. This correlates with pastes which do not turn green. Another important quantity, the weight loss on calcination (LOC) of 28-30% has been selected to represent the confused historic statements of "calcine sufficient to decarbonate", "just enough to minimize the weight" and "underfire to economize on grounding" found in the literature, although not well documented. The analytical approximations for these two constraints have also been obtained using the GP methodology.

The formulation of the optimization problem is as follows:

Strength at one week	$\rightarrow max$
Rate of strength enhancement	$\rightarrow max$
Subject to:	Silica $\leq$ 50
	28% $\leq$ LOC $\leq$ 30%

To find the Pareto-optimal set, first the objective and constraint functions have been plotted to identify the feasible solution domains. Second, a series of optimization runs has been performed by the SQP algorithm at regular intervals. The final solution is represented in Figure 29. The shaded area defines the feasible solution domain, the curves indicate constant levels of strength at one week, the rate of strength enhancement and the constraint functions at their limiting levels. The circled points define the discrete approximation of the Pareto-optimal set.

The analysis of the optimization results (Figure 29) reveals that there are three main zones of study according to the obtained Pareto-optimal set: the upper, middle and lower zones.

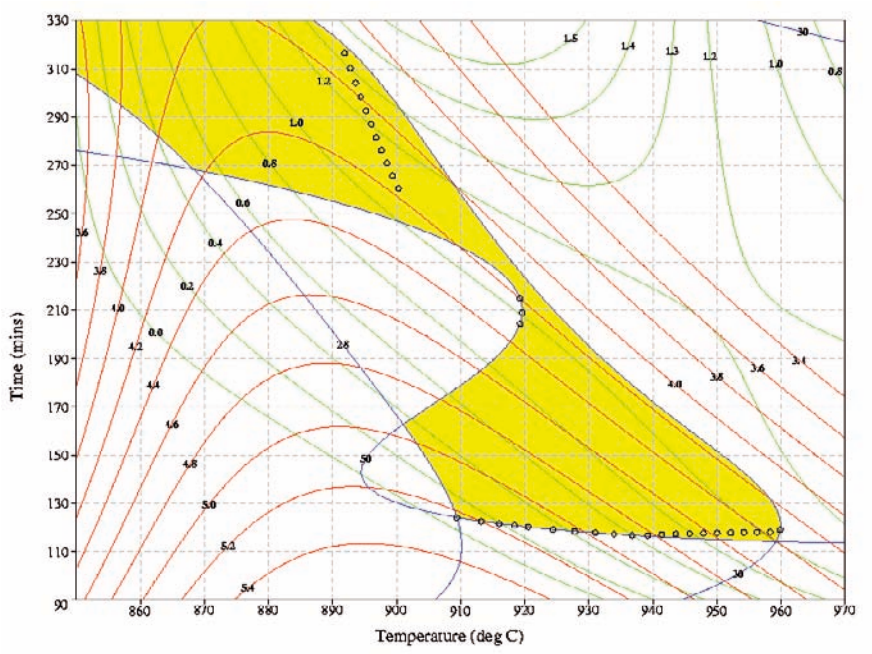


Figure 29. Results of optimization

The lower zone has an essentially constant time band. As temperature increases, the strength at one week decreases but the rate of strength enhancement increases. The upper zone has a narrower temperature band than the lower zone. Here as the temperature reduces, the times increases. As this progression is followed, the same trend as before is noted in terms of the strength at 1 week and the rate of strength enhacement. The middle zone is located in a very narrow area of the feasible domain that suggests uncertainty about the validity of these solutions.

It is possible to identify three points which yield similarly good performance but obtained under different calcination conditions as described in Table 7. It appears that as the calcination temperature is raised, then the residence time is reduced. The final selection of any these calcination conditions will be heavily influenced by energy and hence financial and ecological considerations.

Table 7. Optimal solutions

	Strength at 1 week (*MPa*)	Rate of strength enhancement (*MPa*/week)	Temperature (°*C*)	Time (min)
1	4.4	1.0	957	120
2	4.3	1.0	920	205
3	4.2	1.1	900	260

8 Empirical Modelling of Shear Strength of Reinforced Deep Beams by Genetic Programming (Ashour et al. 2003)

8.1 Introduction

Reinforced concrete (RC) deep beams are characterised as being relatively short and deep, having a thickness that is small relative to their span or depth, and being primarily loaded in their own plane. They are sometimes used for load distribution, for example as transfer girders, pile caps, folded plates and foundation walls. The transition from reinforced concrete shallow beam behaviour to that of deep beams is imprecise. For example, while the ACI code (ACI, 1999), CEB-FIP model code (CEB-FIP, 1993) and CIRIA Guide 2 (CIRIA, 1977) use the span/depth ratio limit to define RC deep beams, the Canadian code (CSA, 1994) employs the concept of shear span/depth ratio. ACI defines beams with clear span to effective depth ratios less than 5 as deep beams, whereas CEB-FIP model code treats simply supported and continuous beams of span/depth ratios less than 2 and 2.5, respectively, as deep beams.

Several possible modes of failure of deep beams have been identified from physical tests but due to their geometrical dimensions shear strength appears to control their design. Despite of the large amount of research carried out over the last century, there is no agreed rational procedure to predict the shear strength of reinforced concrete deep beams (Kong, 1990; Regan, 1993). This is mainly because of the very complex mechanism associated with the shear failure of reinforced concrete beams.

The design of reinforced concrete deep beams has not yet been covered by the British code of practice BS8110 (BSI, 1997) that explicitly states, "for the design of deep beams, reference should be made to specialist literature". Comparisons between test results and predictions from other codes, such as ACI and CIRIA Guide 2, show poor agreement (Tan et al., 1997; Teng et al., 1998).

In this section, the genetic programming (GP) method is used to build an empirical model to estimate the shear strength of reinforced concrete deep beams

subjected to two point loads (Ashour et al. 2003). The GP model will directly evolve from a set of experimental results available in the literature. A parametric study is conducted to examine the validity of the GP model predictions.

8.2 Parameters affecting shear strength of deep beams

Figure 31 shows the geometrical dimensions and reinforcement of a typical reinforced concrete deep beam tested under two point loads. The main parameters influencing the shear strength of reinforced concrete deep beams are the concrete compressive strength, main longitudinal top and bottom steel reinforcement, horizontal and vertical web steel reinforcement, beam width and depth, shear-span and beam-span (Ashour, 2000; Kong et al., 1970; Smith and Vantsiotis, 1982). Those parameters can be expressed in normalised form as follows:

- Normalised shear strength $\lambda = P/b\,h\,f_c'$, where P = shear failure load, b = beam width, h = overall beam depth, f_c' = concrete compressive strength;
- Shear span to depth ratio $x_1 = a/h$;
- Beam span to depth ratio $x_2 = L/h$;
- Smeared vertical web reinforcement ratio $x_3 = A_{sv}\,f_{yv}/b\,s_v\,f_c'$, where A_{sv} = area of vertical web reinforcement, s_v = horizontal spacing of vertical web reinforcement, f_{yv} = yield stress of vertical web reinforcement;
- Smeared horizontal web reinforcement ratio $x_4 = A_{sh}\,f_{yh}/b\,s_h\,f_c'$, where A_{sh} = area of horizontal web reinforcement, s_h = vertical spacing of horizontal web reinforcement, f_{yh} = yield stress of horizontal web reinforcement;
- Main longitudinal bottom reinforcement ratio $x_5 = A_{sb}\,f_{yb}/b\,h\,f_c'$, where A_{sb} = area of main longitudinal bottom reinforcement, f_{yb} = yield stress of main longitudinal bottom reinforcement;
- Main longitudinal top reinforcement ratio $x_6 = A_{st}\,f_{yt}/b\,h\,f_c'$, where A_{st} = area of main longitudinal top reinforcement, f_{yt} = yield stress of main longitudinal top reinforcement.

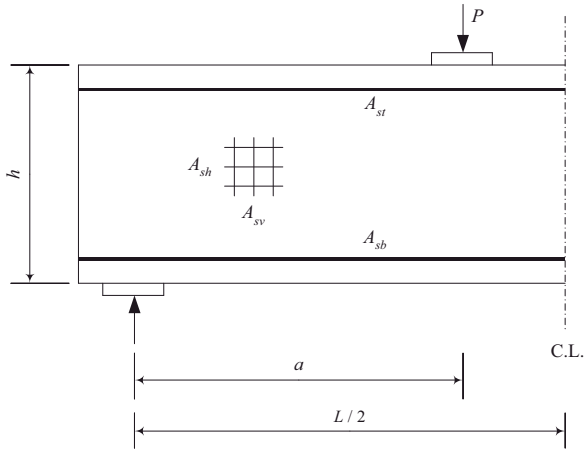


Figure 30. Geometrical dimensions of a reinforced concrete deep beam

Following normal practice established in the majority of papers published on this topic (Mau and Hsu, 1987; Wang et al., 1993), the transformation of the physical variables into dimensionless parameters allowed the reduction of the initial set of 16 variables to only 6 dimensionless variables. A dimensionless format is typically used in the codes of practice and can be easily understood by design engineers. In addition, the dimensionless transformation of the initial physical variables has not been done arbitrarily but follows design expertise of structural engineers, i.e. $x_1=a/h$ could have been defined as $x_1=a/L$ but this would not make sense to a design engineer; x_3 and x_4 define the smeared intensity of vertical and horizontal web reinforcement. In applications where the number of variables is small, the response function produced by GP could be directly related to the physical variables, as suggested by Keijzer and Babovic (1999, 2000) who developed a dimensionally aware GP.

The shear span to depth ratio x_1 is one of the main parameters influencing shear behaviour (Ashour, 2000; Manuel et al., 1971; Paiva and Siess, 1965; Smith and Vantsiotis, 1982). A marked increase in the shear strength occurs in reinforced concrete beams with reducing the shear span to depth ratio. The type of web reinforcement affects the shear strength of reinforced concrete deep beams (Kong et al., 1970; Rogowsky et al., 1986). Most codes of practice provide formulae to calculate the shear strength in which the contribution of the horizontal web reinforcement is higher than that of the vertical web reinforcement. Leonhardt and Walther (1970) suggested that the shear strength of deep beams cannot be improved by the addition of web reinforcement. However, Kong et al. (1970) suggested that improvement is possible to a limited extend.

Rogowsky et al. (1986) concluded that horizontal web reinforcement had no effect on the shear strength while the vertical web reinforcement had a significant influence. As explained above, there is strong disagreement on the influence of web reinforcement on the shear strength of deep beams, in particular the relative effectiveness of vertical and horizontal reinforcement. Although most test results of reinforced concrete deep beams suggest that the span to depth ratio x_2 has very little influence on shear strength (Mau and Hsu, 1987; Subedi, 1988; Wang et al., 1993), most codes of practice use this parameter to define deep beams. In this section, the span to depth ratio will be represented in the GP model as one variable and its effect on the shear strength will be studied. Using the current technique, it will be possible to study the effect of all parameters on the ultimate shear strength of deep beams using all test results available in the literature at the same time; this may eliminate the inconsistency and conflicting conclusions drawn by different researchers.

8.3 Empirical model obtained by GP

There is a large number of test results of reinforced concrete deep beams referred to in the literature. Test results of 141 deep beams reported in (Kong et al., 1970; Kong et al., 1972; Manuel et al., 1971; Paiva and Siess, 1965; Ramakrishnan and Ananthanarayana, 1968; Rogowsky et al., 1986; Smith and Vantsiotis, 1982; Subedi et al., 1986; Suter and Manuel, 1971; Tan and Lu, 1999) are used here to create the GP response. The training data set covers a wide range of each parameter as given in Table 8. All selected beams were tested under two point loads; other load arrangements have been excluded.

Table 8. Range of normalised function and parameters of the training data set

	Minimum	Maximum
x_1	0.28	2.0
x_2	0.9	4.14
x_3	0.0	0.32
x_4	0.0	0.21
x_5	0.023	0.445
x_6	0.0	0.128
$\lambda(x)$	0.029	0.308

The mathematical operators addition, multiplication, division, square and negation and a population size of 500 individuals were selected in the initial runs.

For simplicity of the GP-evolved expression, the power of variables was restricted to positive integer values.

From the beginning, it was observed that the variable x_2 (beam span to depth ratio) had small influence on the shear strength λ and, on one occasion, GP did not include this variable in the evolved expression.

To confirm this observation, several GP runs were undertaken with the fitness function given in equation (7) replaced by the statistical concept of correlation (McKay et al., 1996) as defined in equation (13) below:

$$Q(S_i) = \left| \frac{\sum_{p=1}^{P} \left(\tilde{F}_p - \tilde{\bar{F}} \right) \left(F_p - \bar{F} \right)}{\sqrt{\sum_{p=1}^{P} \left(\tilde{F}_p - \tilde{\bar{F}} \right)^2} \sqrt{\sum_{p=1}^{P} \left(F_p - \bar{F} \right)^2}} \right|, \quad 0 \le Q(S_i) \le 1 \tag{13}$$

where, for a given tree, $\tilde{\bar{F}}$ is the mean of GP predicted function values over the P points in the experimental data, and, similarly, $\bar{F}$ is the mean of the experimental shear strength values over all experimental data. The fitness function given in (13) determines and quantifies the correlation between the independent variables (x_1, x_2, x_3, x_4, x_5, x_6) and the dependant variable λ. The closer the fitness value to 1, the stronger the correlation. In all GP runs, the fitness value Q(Si) in (13) was close to 1 when variable x_2 was not included in the final GP expression. The small relevance of x_2 on the shear strength has also been experimentally observed by other researchers (Ashour, 2000; Kong et al., 1972; Subedi, 1988; Tan et al., 1997; Wang et al., 1993).

In the next stage, only variables x_1, x_3, x_4, x_5 and x_6 were used. Several runs were performed and the solutions analysed on the basis of the simplest generated model that conformed as closely as possible to the engineering understanding of the failure mechanism. When the population size was increased to 1000 individuals and the mutation rate set to 0.001, the following model emerged:

$$\lambda = x_5 * (4.31 + 0.15 * x_1^2 + 12.11 * x_1 * x_5 + 3.34 * x_1 * x_6 + 0.66 * x_3 + 0.47 * x_4 + 23.27 * x_5^2 - 16.97 * x_1 * x_5^2 - 18.22 * x_5 - 2.70 * x_1) \tag{14}$$

Solutions with better fitness than (14) were produced, but they were rejected because of their excessive length. Simplicity is a requirement and, as the complexity of the model increases, its ability to generalise can be affected by the risk of overfitting the data.

The structure of expression (14) was found acceptable, but the coefficients needed to be adjusted in order to satisfy some constraints derived from the engi-

neering knowledge of the problem, such as that the shear strength should be positive for the range of shear span to depth ratio studied. A Sequential Quadratic Programming (SQP) algorithm (Madsen and Tingleff, 1990) was applied to improve the coefficients of equation (15) resulting in the following expression:

$$\lambda = x_5 * (3.50 + 0.20 * x_1^2 + 3.3 * x_1 * x_5 + 3.37 * x_1 * x_6 + 0.63 * x_3 + 0.71 * x_4 + 9.8 * x_5^2 - 1.66 * x_1 * x_5^2 - 10.67 * x_5 - 1.76 * x_1) \quad (15)$$

Further studies with GP and manual postprocessing to adjust the coefficients produced by the SQP algorithm have suggested a simplified final expression as follows:

$$\lambda = A * x_5^2 + B * x_5 + C$$

where $A = -4.56 + 1.68 * x_1$

$B = 2.45 + 0.1 * x_1^2 - 1.16 * x_1 + 3.12 * x_6$ (16)

$C = 0.3 * x_3 + 0.3 * x_4$

It appeared that the variables x_1 (shear span to depth ratio) and x_5 (main longitudinal bottom reinforcement ratio) were the most significant parameters. Alternative expressions with an additional term $x_1*x_6*x_5$ were found, but no relationship between these variables is available as a criterion for the choice between different acceptable expressions. In the literature there is no consensus about the effect of the main longitudinal top reinforcement (represented by x_6 in the above expression) on the shear strength; this requires further investigation and, following that, better understanding of its effect can be reflected in the GP prediction. The web reinforcement contribution (represented by x_3 and x_4) as given by expression (16) is very small.

Expression (16) gives a root mean square (RMS) error over the training data of 0.033. The average ratio between the predicted and experimental shear strength is 1.008, and the standard deviation is 0.23. Figure 31 shows a comparison between the experimental and predicted shear strengths for the training data. To validate the model, 15 additional experimental results, which were not exposed to GP in the training process, were used. The average and standard deviation of the ratio between the predicted and experimental shear strengths are 1.11 and 0.21, respectively. The RMS error over the validation data is 0.035.

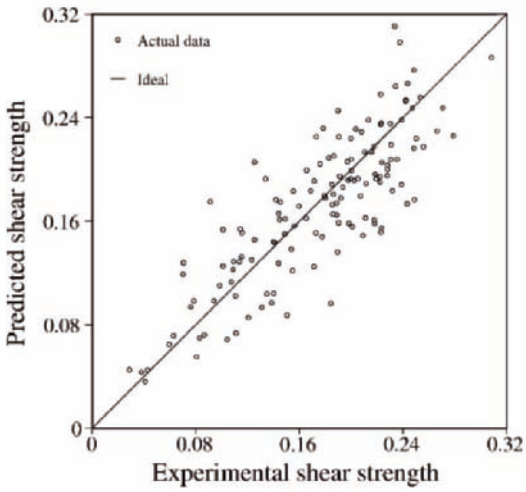


Figure 32. Comparison of experimental and predicted shear strengths

8.4 Conclusions

An empirical model to predict the shear strength of reinforced concrete deep beams has been obtained by GP. Experimental results are used to build and validate the model. Good agreement between the model predictions and experiments has been achieved. As more experimental results and knowledge of the shear behaviour of deep beams become available, the GP prediction could be improved.

The GP model predicts the following behaviour between the shear strength and the influencing parameters:

- The shear span to depth and main longitudinal bottom reinforcement ratios have the most significant effect on the shear strength of reinforced concrete deep beams.
- The shear strength is inversely proportional to the shear span to depth ratio; the higher the shear span to depth ratio, the less the shear strength.
- The shear strength increases with the increase of the main longitudinal bottom reinforcement ratio up to a certain limit beyond which no improvement can be achieved.
- The effect of the beam span to depth ratio and web reinforcement on the shear strength is very small.

References

ACI Committee 318 (1999). *Building Code Requirements for Structural Concrete* (ACI 318-99) *and Commentary* (ACI 318R-99). American Concrete Institute, Detroit.

Aguilar G., Matamoros A.B., Parra-Montesinos G.J., Ramírez J.A. and Wight J.K. (2002). Experimental evaluation of design procedures for shear strength of deep reinforced concrete beams. *ACI Structural Journal* 99:539-548.

Altair OptiStruct version 4.0 (2000). Altair Engineering Inc.

Alvarez L.F., Hughes D.C. and Toropov,V.V. (2000). Optimization of the manufacturing process for Roman cement. In Sienz, J. (ed.), *Engineering Design Optimization. Process and Product Improvement. Proceedings of 2nd ASMO UK / ISSMO Conference*, 27-32.

Ashour A.F. (2000). Shear capacity of reinforced concrete deep beams. *Structural Engineering Journal ASCE* 126:1045-1052.

Ashour A.F., Alvarez L.F. and Toropov V.V. (2003). Empirical modelling of shear strength of RC deep beams by genetic programming. *Computers and Structures* 81:331-338.

Audze P. and Eglais V. (1977). New approach for planing out of experiments. *Problems of Dynamics and Strengths* 35:104-107. Zinatne Publishing House, Riga

Baruch M. (1982). 15 Optimal correction of mass and stiffness matrices using measured modes. *AIAA J.* 20:441.

Bates S.J., Sienz J. and Toropov V.V. (2004). Formulation of the optimal Latin hypercube design of experiments using a permutation genetic algorithm, *Proceedings of. 45th AIAA/ASME/ASCE/AHS/ ASC Structures, Structural Dynamics & Materials Conf.* Palm Springs, California, 19-22 April 2004.

Booker A.J. (1998). Design and analysis of computer experiments. AIAA-98-4757. In *Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* AIAA. Part 1:118-128. St. Louis.

Box G.E.P. and Draper N.R. (1987). *Empirical model-building and response surfaces.* New York, John Wiley and Sons.

British Standards Institution (1997). *Structural use of concrete*, BSI, BS8110: Part 1.Milton Keynes,

CEB-FIP (1993). *Model Code 1990 for concrete structures*. Thomas Telford Services, Ltd.. London, Comité Euro-International du Béton, Laussane.

Choi K.K., Youn B.D. and Yang R.-J. (2001). Moving Least Square Method for reliability-based design optimization. *Proc. 4th World Cong. Structural & Multidisciplinary Optimization*, Dalian, China

CIRIA Guide 2 (1977). *The Design of Deep Beams in Reinforced Concrete*. Over Arup and Partners, and Construction Industry Research and Information Association, London, reprinted 1984.

CSA Technical Committee on Reinforced Concrete Design, A23.3-94 (1994). *Design of Concrete Structures.* Canadian Standards Association, Ontario, Canada.

Davis L. (1985). Applying adaptive algorithms to epistatic domains, *Proc. Int. Joint conf. on Artificial Intelligence* 162-164.

De Paiva H. A. and Siess C. P. (1965). Strength and behaviour of deep beams in shear. *Journal of the Structural Division, ASCE* ST5:19-41.

FIA (2002). *Article 3: Bodywork and Dimensions*: Subsection 3.4.2, 2002. Formula One Technical Regulations.

Friedman J.H. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics* 19:1.

Goldberg D.E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, Reading, MA.

Goldberg, D.E. and Lingle, R. (1985). Alleles, loci, and the TSP, *Proc. 1st Int. conf. on Genetic Algorithms*, 154-159. Hillsdale, NJ.

Cawley P. and Adams R.D. (1979). The location of defects in structures from measurements of natural frequencies. *J. Strain Analysis* 14:49-?.

Gray G. Li Y., Murray-Smith D. and Sharman K. (1996). Structural system identification using genetic programming and a block diagram oriented simulation tool. *Electronics Letters* 32:1422-1424-?.

Gray G., Murray-Smith D. Sharman K. and Li,Y. (1996). Nonlinear model structure identification using genetic programming. In *Late-breaking papers of Genetic Programming '96*, Stanford, CA.

Gürdal Z. Haftka R.T. and Hajela, P. (1999). *Design and optimization of laminated composite material.* John Wiley & Sons.

Haftka R.T. (1997). Optimization and experiments – a survey. In Tatsumi T., Watanabe E. and Kambe T., eds., *Theoretical and Applied Mechanics 1996, Proceedings of XIX International Congress of Theoretical and Applied Mechanics,*303-321. Elsevier.

Hajela P. and Soeiro F. J. (1990a). Structural damage detection based on static and modal analysis. *AIAA J.* 28:1110-?.

Hajela P. and Soeiro F.J. (1990b). Recent Developments in Damage Detection Based on System Identification Methods. *Structural Optimization* 2:1-?.

Hassiotis S. and Jeong G.D. (1993). Assessment of structural damage from natural frequency measurements. *J. Computers & Structures* 49:679-?.

Iman R.L. and Conover W.J. (1980). Small sample sensitivity analysis techniques for computer models, with an application to risk assessment, *Communications in Statistics, Part A. Theory and Methods* 17:1749-1842.

Jin R., Chen W. and Sudjianto A. (2003). An efficient algorithm for constructing optimal design of computer experiments. DETC-DAC48760, *2003 ASME Design Automation Conference*, September 2-6, 2003. Chicago, IL.

Johnson M., Moore L. and Ylvisaker D. (1990). Minimax and maximin distance designs. *J. Statist. Plann. Inference* 26:131-148.

Kabe A.M. (1985). Stiffness matrix adjustment using mode data. *AIAA J* 23:1431.

Keane A. and Nair P. (2005). *Computational Approaches for Aerospace Design: The Pursuit of Excellence*. Wiley.

Keijzer M. and Babovic V. (1999). Dimensionally aware genetic programming. Orlando. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M. and Smith, R.E, eds., *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, Florida, USA. Morgan Kauffman Publishers.

Keijzer M. and Babovic V. (2000). Genetic programming within a framework of computer-aided discovery of scientific knowledge. In: Whitley D., Goldberg D. Cantu-Paz E., Spector L., Parmee I. and Beyer H., eds., *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, Las Vegas, USA. Morgan Kauffman Publishers.

Kinnear, K.E. (1994). *Advances in genetic programming*. MIT Press.

Kong F.K., Robins P.J. and Cole D. F. (1970). Web reinforcement effects on deep beams. *ACI Journal* 67:1010-1017.

Kong F. K., Robins P. J., Kirby D.P. and Short D.R. (1972). Deep beams with inclined web reinforcement. *ACI Journal* 69:172-176.

Kong F. K. (1990). *Reinforced concrete deep beams*. Edited by F.K. Kong; Blackie. New York.

Koza J.R. (1992). *Genetic Programming: On the programming of computers by means of natural selection*. MIT Press.

Lapierre H. and Ostiguy G. (1990). Structural model verification with linear quadratic optimization theory. *AIAA J.* 28:1497-?.

Leonhardt F., and Walther R. (1970). *Deep beams*. Deutscher Ausschuss Für Stahlbeton Bulletin 178. Wilhelm Ernst and Sohn (Berlin), CIRIA English Translation.

Mackay M.D. , Beckman R.J. and Conover W.J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21:239-245.

Madsen K. and Hegelund P. (1991). *Non-gradient subroutines for non-linear optimization. Institute for Numerical Analysis*, Technical University of Denmark, Report NI-91-05.

Manuel R.F. Slight B.W. and Suter G. T. (1971). Deep beam behaviour affected by length and shear span variations. *ACI Journal* 68:954-958.

Mau S.T. and Hsu T.T.C. (1987). Shear strength prediction for deep beams with web reinforcement. *ACI Structural Journal* 513-523.

McKay B., Willis M.J., Hidden H.G., Montague G.A. and Barton G.W. (1996). Identification of industrial processes using genetic programming. In Friswell, M.I., and Mottershead, J.E., eds., *Proceedings of International Conference on Identification in Engineering Systems*, Swansea, 328-337. The Cromwell Press Ltd.

Matheron G. (1963). Principles of geostatistics. *Economic Geology* 58:1246-1266.

Michalewicz Z. (1992). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag

Myers R.H. and Montgomery D.C. (1976). *Response surface methodology: Process and Product Optimization Using Designed Experiments*. John Wiley & Sons, New York, NY.

Nevey S. and Stephens M. (2000). Weight reduction in a Formula One composite wing. *Altair Hyperworks: second UK conference*. Towcester Racecourse, UK.

Oliver I.M., Smith D.J. and Holland, J.R.C. (1987). A study of permutation crossover operators on the travelling salesman problem. *Proc. 2nd Int. Conf. on Genetic Algorithms*, 411-423, Massachusetts Institute of Technology, Cambridge, MA.

Ramakrishnan V. and Ananthanarayana Y. (1968). Ultimate Strength of deep beams in shear. *ACI Journal* 65:87-98.

Ravaii H., Toropov V.V. and Horoshenkov K.V. (1997). Structural damage recognition based on optimization techniques. In Gutkowski, W., Mroz, Z., eds., *Proc. of 2nd World Congress of Structural and Multidisciplinary Optimization,* Zakopane, Poland, May 1997, 1:299-304. Polish Academy of Sciences.

Ravaii H., Toropov V.V. and Horoshenkov K.V. (1998a). Mixed numerical-experimental damage recognition in steel structures. In: Allison I.M. (ed.), *Experimental Mechanics. Advances in Design, Testing and Analysis - Proc. 11th Int. Conf. on Experimental Mechanics*, 1: 77-82,. Oxford, August 24-28, 1998. A.A. Balkema, Rotterdam,

Ravaii H., Toropov V.V. and Mahfouz S.Y. (1998b). Application of a genetic algorithm and derivative-based techniques to identification of damage in steel structures. In: M. Tanaka, G.S. Dulikravich, (eds), *Inverse Problems in Engineering Mechanics - Proc. Int. Symposium*, 571-580, Nagano City, Japan, March 24-27, 1998. Elsevier.

Regan P.E. (1993). Research on shear: a benefit to humanity or a waste of time?. *The Structural Engineer* 71:337-346.

Rogowsky D. M., MacGregor J.G., and Ong S. Y. (1986). Test of reinforced concrete deep beams. *ACI Structural Journal* 83:614-623.

Roux W.J., Stander N. and Haftka R.T. (1998). Response surface approximations for structural optimization. *International Journal for Numerical Methods in Engineering* 42:517-534.

Sacks, J., Schiller, S.B. and Welch, W.J. (1989). Designs for computer experiments, *Technometrics*, 34:15-25.

Sahab M.G., Toropov V.V. and Ashour A.F. (2001). Cost optimization of reinforced concrete flat slab buildings by a genetic algorithm. *4th World Congress of Structural an Multidisciplinary Optimization*. Dalian, China.

Shewry M. and Wynn H. (1987). Maximum entropy design. *J. Appl. Statist*. 14:165-170.

Smith K. N. and Vantsiotis A. S. (1982). Shear strength of deep beams. *ACI Structural Journal* 79:201-213.

Stephens M., Gambling M., Jones R.D., Toropov V.V. and Alvarez L.F. (2002). Weight optimization of a formula one car composite component. In: P.D. Gosling (ed.), *Proc. of 4th ASMO UK / ISSMO Conf. on Engineering Design Optimization. Process and Product Improvement*, July 4-5, 2002, 178-183. Newcastle

Subedi N. K., Vardy A.E. and Kubota N. (1986). Reinforced concrete deep beams- some test results. *Magazine of Concrete Research* 38:206-219.

Subedi N.K. (1988). Reinforced concrete deep beams: a method of analysis. *Proceedings of Institution of Civil Engineering*, Part 2, 85:1-30.

Suter G. T. and Manuel, R. F. (1971). Diagonal crack width control in short beam. *ACI Journal* 68:451-455.

Tan K.H., Weng L. W. and Teng, S. (1997). A strut-and-tie model for deep beams subjected to combined top-and-bottom loading. *The Structural Engineer Journal* (London), 75:215-225.

Tan K. H. and Lu H. Y. (1999). Shear behaviour of large reinforced concrete deep beams and code comparisons. *ACI Structural Journal* 96:836-845.

Teng S., Kong F. K. and Poh S. P. (1998). Shear strength of reinforced and pre-stressed concrete deep beams. Part I: Current design methods and a proposed equation. *Proceedings Institution of Civil Engineers, Structures & Buildings* 128:112-123.

Toropov V.V. (2001). Modelling and approximation strategies in optimization - global and mid-range approximations, response surface methods, genetic programming, low / high fidelity models. In: Blachut J.; Eschenauer H. A. (eds), *Emerging Methods for Multidisciplinary Optimization,* CISM Courses and Lectures, No. 425, International Centre for Mechanical Sciences, 205-256. Springer.

Toropov V. V. and Alvarez L.F. (1998a). Application of genetic programming and response surface methodology to optimization and inverse problems. In: M. Tanaka, G.S. Dulikravich (eds), *Inverse Problems in Engineering Mechanics - Proc. Int. Symp.*, 551-560, Nagano City, March 1998. Elsevier,.

Toropov V. V. and Alvarez L.F. (1998b). Approximation model building for design optimization using genetic programming methodology. *Paper AIAA-98-4769, 7th AIAA/USAF/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization,* 490-498, St. Louis, USA, September 1998.

Toropov V. V., Alvarez L.F. and Ravaii, H. (1999a). Structural damage recognition using optimization techniques and genetic programming methodology, *3rd ISSMO/UBCAD/UB/AIAA World Congress of Structural and Multidisciplinary Optimization*, (CD Proc.), Buffalo, NY, USA, May 17-21, 1999.

Toropov V. V., Alvarez L.F. and Ravaii H. (1999b). Recognition of damage in steel structures using genetic programming methodology. In: M.I. Friswell, J.E. Mottershead, A.W. Lees (eds.), *Proc. 2nd International Conf. on Identification in Engineering Systems*, 382-391.Swansea, March 29-31, 1999. The Cromwell Press Ltd.

Toropov V.V., Bates S.J. and Querin O.M. (2007). Generation of extended uniform Latin hypercube designs of experiments. 9*th Int. Conference on the Application of Artificial Intelligence to Civil, Structural and Environmental Engineering*. St. Julians, Malta, 18-21 September 2007.

Toropov V.V. and Markine V.L. (1996). The use of simplified numerical models as mid-range approximations. *6th AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization*, Part 2, 952-958. Bellevue, WA.

Toropov V.V., Markine V.L., and Holden C.M.E. (1999). Use of mid-range approximations for optimization problems with functions of domain-dependent calculability, *Proceedings of 3rd World Congress of Structural and Multidisciplinary Optimization,* Buffalo, NY.

Toropov V.V., Schramm U., Sahai,A., Jones R. and Zeguer, T. (2005). Design optimization and stochastic analysis based on the Moving Least Squares Method. In: Herskovits J, Mazorche S. and A. Canelas(eds) *6th World Congress of Structural and Multidisciplinary Optimization.* Paper 9412. CD-ROM Proceedings. Rio de Janeiro, Brazil, May 2005.

Toropov V.V. and Yoshida F. (2005). Application of advanced optimization techniques to parameter and damage identification problems. In: Mroz, Z. and Stavroulakis, G.E. (eds.), *Parameter Identification of Materials and Structures, CISM Courses and Lectures,* Vol. 469:177-263, International Centre for Mechanical Sciences. Springer.

Wang W., Jiang D.-H., and Hsu C.-T.T. (1993). Shear strength of reinforced concrete deep beams. *Journal of Structural Engineering* 119:2294-2312.

CHAPTER 4

Advances in Neural Networks in Computational Mechanics and Engineering

Jamshid Ghaboussi

Department of Civil and Environmental Engineering
University of Illinois at Urbana Champaign
Urbana, Illinois 61801, USA

Abstract. A vast majority of engineering problems are inverse problems, while the mathematically based engineering problem solving methods and computational mechanics are primarily capable of solving forward problems. Nature has evolved highly effective, robust and imprecision tolerant problem solving strategies for very difficult inverse problems. Biologically inspired soft computing methods such as neural network, genetic algorithm and fuzzy logic inherit the basic characteristics of nature's problem solving methods and, as such, they are suitable for inverse problems in engineering. In this chapter we start by discussing the fundamental differences between the mathematically based engineering problem solving methods and biologically inspired soft computing methods. Bulk of the rest of the chapter is devoted to applications of neural networks in computational mechanics and several difficult inverse engineering problems.

1 Introduction

The first application of neural networks in computational mechanics and structural engineering was in the modeling of the constitutive behavior of materials (see Ghaboussi Garrett and Wu, 1990, 1991). In that application it was demonstrated that neural networks are capable of learning the constitutive behavior of plane concrete in two-dimensions directly from experimental results. Although, that work demonstrated the potential of neural networks in computational mechanics, there remained a number of important unresolved questions. Some of these questions dealt with the theoretical issues such as: how do the neural networks learn and what do they learn from training data; what are the limitations of such learning in computational mechanics and how can these limitations be quantified; how to determine the sufficiency of data for training neural networks; and other similar issues. For example, in the first application of neural networks in constitutive modeling (see Ghaboussi Garrett and Wu 1990, 1991) the term "comprehensive data set for training of neural networks" was

used. However, at that time it was not clear how to determine what constituted a comprehensive data set. Other unresolved questions dealt with practical issues, such as the size of neural networks and training algorithms. Considerable advances have been made in understanding these issues and some of these advances will be discussed in this chapter.

It is assumed that the reader has a basic knowledge of neural networks and computational mechanics. The discussions in this chapter are devoted to the issues that make the application of neural networks in computational mechanics and engineering possible and effective, leading to new and robust methods of problem solving. Since neural networks and other computational intelligence or soft computing tools are biologically inspired, we will first examine the computing and problem solving strategies in nature. We will discuss the fundamentals of the differences between the computing and problem solving strategies in nature and our mathematically based problem solving methods such as computational mechanics. Understanding these differences plays an important role in development and utilization of the full potential of soft computing methods. They also help us understand the limitations of soft computing methods.

Another important focus of this chapter is on examining the characteristics of the problems that biological systems encounter and how nature has evolved effective strategies in solving these problems. A vast majority of these problems are inverse problems, as are a vast majority of engineering problems, including problems in mechanics. Again, our problem solving strategies have evolved differently as they are based on mathematics. We will examine the limitations of the mathematically based methods in directly addressing the inverse problems. Biologically inspired soft computing methods offer potentials for addressing these methods. However, full utilization of these potentials requires new thinking and full understanding of the fundamental issues that will be discussed in this chapter.

2 Computing and Problem Solving in Nature

Computing and problem solving does occur in nature in a massive way. We normally do not assign problem solving capabilities to biological systems, such as animals, eco-systems and insect colonies. Computing in nature does occur in human and animal brains. Although brains are very different than our computers, they nevertheless do perform computing tasks. Moreover, human or animal brains perform computing tasks that are far more complex and well beyond the capabilities of our present computers. Computing and problem solving strategies in nature have evolved to develop capabilities that are important in the survival and propagation of the species. Exploring these capabilities is important in understanding the fundamentals of computing and problem solving in nature; these fundamentals are very different than the

fundamentals underlying our engineering problem solving methodology, including mechanics, computation and computational mechanics. Understanding these differences is essential in order to effectively use the biologically inspired soft computing methods in engineering and computational mechanics.

Computing and problem solving strategies in nature have evolved to solve difficult inverse problems. Most of the problems that human and animal brains solve are inverse problems. Our engineering problem solving methods work differently. Although most engineering problems are also inverse problems, we normally pose them as forward problems and solve them that way. In a forward problem the system (or model of the system) and the input to the system is known, and the output needs to be determined. We pose and solve the engineering problems as forward problems because the mathematically based problem solving methods in mechanics and engineering are only suitable for solving forward problems. These issues will be further discussed later in this chapter.

There are two types of inverse problems. In the first type of inverse problems the system and the output of the system is known and the input to the system that produces the known output needs to be determined. In the second type of inverse problems the input and output to the system are known and the system needs to be identified.

One of the important classes in the first type of inverse problems in nature is recognition, such as: voice recognition, face recognition, recognizing the prey, recognizing danger, etc. The forward problems in this class are when we know an individual and hear his/her voice and see his/her face. The inverse problem is when we hear a voice or see a face and have to identify the person. Another class of inverse problems in nature is control; our brains control our limbs or control the functioning of internal organs. The forward problem in this case is to send a known signal to a limb or an organ and observe the response of the limb or organ. The inverse problem is to determine the right input signal that would produce the desired response in a limb or an organ.

Brains have evolved to perform these inverse tasks with two important characteristics. These inverse problems have to be solved robustly and in real-time. Survival depends on these characteristics. Animals have to recognize danger instantaneously in order to take evasive action. Similarly, they have to recognize their prey in real-time to have any chance of hunting it successfully. Delay in recognition can be detrimental to a human being or an animal. Robustness is also a key factor in solving the inverse problems in nature. Images may be partially blocked or voices may contain noise. Robustness in some important way implies imprecision tolerance. All of the inverse problems in nature have a certain degree of imprecision tolerance. For example, a predator needs to determine its distance to his prey. However, it does not need to know that distance precisely; an approximate value will suffice. Similar degrees of

imprecision tolerance exist in all of the inverse problems in nature. Nature's problem solving strategies have evolved with a degree of imprecision tolerance.

Brains have evolved as massively parallel systems of neurons connected with synapses and dendrites. The massively parallel structure of the brains makes possible the real-time and robust computing in solving the inverse problems. Moreover, it inherently operates with a degree of imprecision tolerance. This is of course very different from our present day computers that have a high degree of precision requirement. We will also discuss later in this chapter why our computers are not suitable for directly solving the inverse problems.

How do biological systems solve inverse problems? This is an important question, since similar methods can be deployed in solving inverse problems in engineering and computational mechanics. Nature and brains solve inverse problems by utilizing two important strategies: learning and reduction in disorder. First, we will discuss the strategy of learning in solving inverse problems. Of course, as we will see later, learning itself is a form of reduction in disorder.

Learning from the forward problems is the main strategy that nature uses to solve the inverse problems. For example, in the problem of voice and face recognition the repeated instances of seeing a person and hearing their voice are the forward problems that lead to learning to solve the inverse problem of recognition. Learning to control ones limbs also occurs by the repetition of the forward problem. This is clearly seen when a baby learns through repetition to control the movement of his or her arms, fingers and other limbs. The information learned from forward problems is stored in the synapses of the neurons in our brains. Similar strategies can be used in engineering and computational mechanics. In this case, the information acquired for solving the inverse engineering problem is stored in the connection weights of the neural networks. Later in this chapter we will present a case of a successful application of the same strategy in solving a difficult inverse problem in engineering. What makes it possible to apply this strategy to engineering problems is the learning capability of the neural networks, where the knowledge and information is stored in the connection weights.

Reduction in disorder is also a powerful method that nature uses for solving inverse problems. This may not be as clearly observable as the learning. In fact, reduction in disorder underlies learning in massively parallel systems, such as brains and neural networks. As the collective state of a complex system (brains and neural networks are complex systems) approaches a solution to a specific problem, disorder in the state corresponding to that problem reduces. Evolution also leads to reduction in disorder: in this case in a self-organized manner. Organisms have evolved a high degree of order (and a very low degree of disorder). Similarly, evolutionary methods, such as genetic algorithm, also lead to reduction in disorder. We can also say that there is an increase in order (or

reduction in disorder) when brains or neural networks approach a state for storing acquired knowledge and information. Acquisition and storing of the information in the synapses or connection weights constitutes learning.

3 Mechanics, Computation and Computational Mechanics

The field of mechanics deals with observing and modeling the natural phenomena. When a physical event is observed - whether it is an experiment or a naturally occurring event - new information is generated. This information about the physical event is contained in the data that may consist of the recordings of the physical event and/or numerical data from the measurements. The next step is to create a model that can be used to describe that event and other similar events, leading to unifying principles. The universal approach, from the inception of mechanics, has been to develop mathematical models of observed phenomena and physical events. We normally accept this approach without even thinking about it, with the implied understanding that the mathematical modeling is the only possible method that can be used to describe the natural phenomena. Here, we need to further examine this approach, since it is not the only possible method of describing the natural phenomena. In fact, there are many other methods. In this chapter we are interested in methods that involve extracting and storing the information about the natural phenomenon contained in the data, in the connection weights of neural networks. Information contained in a mathematical model can also be stored in a neural network. Since developing a mathematical model of a natural phenomenon and storing the information about that phenomenon are two completely different approaches, it is important that we examine the fundamentals of both approaches. We will examine the fundamentals of mathematically based mechanics and computational mechanics in this section,

All modeling, engineering problem solving methods, and computation is based on mathematics. We will refer to them as mathematically based methods. All the mathematically based methods inherit their characteristics from mathematics. Normally, we do not even think about the fundamental characteristics of the mathematically based methods. However, in this case we need to examine them, since they are so radically different from the fundamental characteristics of the problem solving strategies in nature that underlie the soft computing or computational intelligence methods.

The three fundamental properties of the mathematically based methods in engineering and computational mechanics that are inherited from mathematics are: precision; universality; and functional uniqueness.

3.1 Precision

Precision or “exactness” is a property of all the mathematically based methods. It is because of the precision requirement that all computing, including

computational mechanics, is considered as "hard computing". Precision requirements are the most significant differences between the soft computing and hard computing methods. Precision requirements may not be obvious to the users of engineering software. For example, the input to a typical finite element analysis program may define the geometry of the structure and specify the material parameters and loads. These quantities, especially the material parameters, cannot be determined with a high degree of precision. However, the computer software requires precise values. Users of the computer software often provide the best estimates of the input parameters and they do not view them as precise. However, the mathematically based methods and hard computing software consider the input parameters to be precise to within round off (normally six to fourteen significant figures). Similarly, the output of the engineering computer software is also precise to within round off. However, they are not often considered so by the users of the software. In fact, there may not be much use for a high degree of precision in most engineering and computational mechanics problems.

3.2 Universality

Universality is such a basic property of all mathematical functions that we usually do not think about it. Universality means that the mathematical functions are defined and realized universally for all the possible values of their variables. Again, mathematically based methods use functions that are universal but the physical phenomena they describe most often are not. A simple example is the linearly elastic material behavior. When we write the expression that stress = modulus of elasticity x strain, it describes a phenomenon that is only valid over a range of the stresses and strains. But the equation itself is valid for all the possible values of stress and strain. We will see later that the same information can be learned by a neural network to within an imprecision tolerance over the range of its validity.

3.3 Functional uniqueness

Mathematical functions are also unique in the sense that each function provides a unique mapping different from all the other functions. For example, there is only one sin(x) function and it is valid for all the possible values of x from minus infinity to plus infinity. We will see later that in soft computing, different neural networks, with different numbers of neurons can represent the same function over a range of x to within a prescribed level of imprecision tolerance.

The consequences of *precision, universality, and functional uniqueness* in the mathematically based engineering problem solving methods and hard computing are that these methods are only suitable for solving forward problems. In the forward problems the model of the system and the input to that system are known, and the output of the system is determined. Mathematically

based methods are not suitable for directly solving the inverse problems. Although a vast majority of engineering and computational mechanics problems are inherently inverse problems, we do not pose them as such, and do not consider solving them as inverse problems. An important reason for this is that the inverse problems do not have unique solutions, and very often there is not enough information to determine all the possible solutions. The few inverse problems that are tackled with the conventional mathematically based methods are formulated such that the result is determined from the repeated solutions of the forward problems.

4 Hard and Soft Computing Methods

Almost all the computing, including computational mechanics and finite element analysis, have to be considered hard computing. All computing are based on mathematical approaches to problem solving, and they inherit their basic characteristics from mathematics.

The hard computing methods often solve an idealized precise problem deterministically. Using the current hard computing methods usually involves three steps. First, the problem is idealized to develop a precise mathematical model of the system and the input data. Next, the hard computing analysis is performed within the machine precision on a sequential or parallel computer. Finally, the output to the idealized problem is obtained with a high degree of precision, often much higher level of precision than required in practice. This form of hard computing method is often used in modeling and evaluation of behavior and design of physical systems that are often associated with a certain level of imprecision and uncertainty. Engineering judgment is used in an ad hoc manner to utilize the results of the hard computing analyses.

The consequence of the mathematically based precision in the hard computing methods is that there is no built-in procedure for dealing with uncertainty, lack of sufficient information, and scatter and noise in the data. All the input data must be provided and, where there is a gap, estimates must be made to provide “reasonable precise values" of the input data. Also, all the noise and scatter must be removed before providing the input data to the hard computing methods. Inherently, there are no internal mechanisms for dealing with the uncertainty, scatter and noise. Consequently, the hard computing methods generally lack robustness.

Hard computing methods are more suitable for direct or forward analysis and are often used for problems posed as direct. It is very difficult to solve inverse problems with the current state of the hard computing methods. As will be discussed in a later section, there are many inverse problems that are either not solved currently, or a simplified version of these problems is posed and solved as direct problems.

4.1 Biologically Inspired Soft Computing Methods

Soft computing methods are the class of methods which have been inspired by the biological computational methods and nature's problem solving strategies. Currently, these methods include a variety of neural networks, evolutionary computational models such as genetic algorithm, and linguistic based methods such as fuzzy logic. These methods are also collectively referred to as Computational Intelligence Methods. These classes of methods inherit their basic properties and capabilities from the computing and problems solving strategies in nature. As mentioned earlier, these basic characteristics are fundamentally different from the basic characteristics of the conventional mathematically based methods in engineering and computation.

In the majority of the applications of neural networks and genetic algorithm, researchers have used these methods in limited ways in simple problems. In the early stages of introduction of any new paradigm, it is initially used in a similar manner as the methods in existence prior to the introduction of the new paradigm. Neural networks and other computational intelligence methods are new paradigms. It can be expected that initially they will be used in similar ways as the mathematically based methods in engineering and computational mechanics, as are the vast majority of current applications. Such applications seldom exploit the full potential of the soft computing methods. For example, neural networks are often used as simple function approximators or to perform tasks that simple regression analysis will suffice. Similarly, genetic algorithm is often used in highly restricted optimization problems. The learning capabilities of neural networks and the powerful search capabilities of genetic algorithm can accomplish far more. They can be used in innovative ways to solve problems which are currently intractable and are beyond the capability of the conventional mathematically based methods. Problem solving strategies of the biological systems in nature often provide good examples of how these methods can be used effectively in engineering. Natural systems and animals routinely solve difficult inverse problems. Many of these problem solving strategies can be applied to engineering problems.

4.2 Imprecision Tolerance and Non-universality

Biological systems have evolved highly robust and effective methods for dealing with the many difficult inverse problems, such as cognition; the solution of these problems is imperative for the survival of the species. A closer examination of cognition will reveal that for animals, precision and universality are of no significance. For example, non-universality in cognition means that we do not need to recognize all the possible voices and faces. Recent studies have provided a plausible explanation for the large size of human brains based on cognition. The large size of human brains have evolved so that the humans can recognize the members in their living groups. Human beings lived in groups of

150 to 200 individuals and our brains had to be large enough to recognize that many individuals. Our brains are only capable of recognizing 150 to 200 faces. This non-universality is in contrast with the mathematically based methods. If it were possible to develop a mathematically based cognition method, it would be universal and it would be able to recognize all the possible faces.

On the other hand, real time response and robustness, including methods for dealing with uncertainty, are essential in nature. Soft computing methods have inherited imprecision tolerance and non-universality from biological systems. For example, a multi-layer, feed-forward neural network representing a functional mapping is only expected to learn that mapping approximately over the range of the variables present in the training data set. Although different levels of approximation can be attained, the approximate nature of the mapping is inherent.

Another inherent characteristic of the soft computing methods is their non-universality. Soft computing methods are inherently designed and intended to be non-universal. For instance, neural networks learn mappings between their input and output vectors from the training data sets and the training data sets only cover the range of input variables which are of practical interest. Neural networks will also function outside that range. However, the results become less and less meaningful as we move away from the range of the input variables covered in the training set.

4.3 Functional Non-uniqueness

A third characteristic of the soft computing methods is that they are functionally non-unique. While mathematical functions are unique, neural network representations are not unique. There is no unique neural network architecture for any given task. Many neural networks, with different numbers of hidden layers and different number of nodes in each layer, can represent the same association to within a satisfactory level of approximation. It can be clearly seen that the functional non-uniqueness of the neural networks is the direct consequence of their imprecision tolerance.

An added attraction of soft computing methods in computational mechanics is as the consequence of the imprecision tolerance and random initial state of the soft computing tools. This introduces a random variability in the model of the mechanical systems, very similar to the random variability which exists in the real systems. The random variability plays an important role in most practical problems, including nonlinear quasi static and dynamic behavior of solids and fluids where bifurcation and turbulence may occur. Finite element models are often idealized models of actual systems and do not contain any geometric or material variability. An artificial random variation of the input parameters is sometimes introduced into the idealized finite element models to account for the

random scatter of those parameters in the real systems. On the other hand, the soft computing methods inherently contain random variability.

In summary, we have discussed the fact that the mathematically based methods in engineering and computational mechanics have the following basic characteristics:

Precision
Universality
Functional uniqueness

In contrast, biologically inspired soft computing methods have the following opposite characteristics:

Imprecision tolerance
Non-universality
Functional non-uniqueness

Although it may be counterintuitive, these characteristics are the reason behind the potential power of the biologically inspired soft computing methods.

5 Neural Networks as Soft Computing Tools

The discussion in this section will be focused on multi-layer, feed-forward neural networks, which are currently the most commonly used neural networks in engineering applications (see Ghaboussi and Wu, 1998, Ghaboussi, 2001). These neural networks consist of several layers of artificial neurons or processing units. The input and the output layers are the first and the last layers. The layers between the input and output layers are referred to as the "hidden layers". Each neuron is connected to all the neurons in the next layer, and the signals propagate from the input layer, through the hidden layers, to the output layer. The number of neurons in the input and output layers are determined by the formulation of the problem. The number of neurons in the hidden layers defines the capacity of the neural network which is related to the complexity of the underlying information in the training data set. The relationship between the number of neurons in the hidden layers, the capacity of neural networks, and the complexity of the underlying process being modeled is not easily quantifiable at the present. The input vector is provided as the activation of the neurons at the input layer. Signals propagate from the input layer, through the hidden layer, to the output layer. The activations of the output nodes constitute the output of the neural network.

A neural network is trained with the training data set, such that within the domain of the training data it approximately represents the mapping between the input and output vectors that exists in the training data set. The approximation in neural networks is represented by the error vectors within the domain of the training data. Training of the neural network is the process of reducing the norm of the error vector to a level below a tolerance.

A neural network learns to satisfy its training data set approximately. It differs fundamentally from a mathematical interpolation function, which matches the training data set exactly. Neural networks are also different from a regression analysis, which requires a specified function whose parameters are determined.

In general, the output errors for the training data depend on a number of factors relating to the complexity of the underlying information present in the training data set, as well as the neural network architecture and the training process. It is important to note that it is not desirable to reduce the error too much. In the limit, if the output error is reduced to zero, then the neural network would be functioning similar to an interpolation function and it will lose its generalization capability between the data points in the training data set. The generalization capability of neural networks is the direct consequence of the imprecision tolerance.

Non-universality of soft computing methods that was discussed earlier is an important consideration in the use of neural networks. Non-universality means that neural networks can only learn to approximate relations that are within the range of input variables present in the training data. Outside that range the response of the neural network is not controlled. This is demonstrated with a simple example shown in Figure 1. In this example a simple neural network NN(1, 3, 3, 1) (the numbers in the parenthesis are the number of node in the input, hidden and output layers) is trained with two sets of data taken from within two ranges of sin(x) shown in the upper portion of the figure. Shown in the lower portion of the figure are the responses of the trained neural networks. It is clearly seen that the two neural networks have learned to approximate sin(x) within the range of x in their training data. Outside those ranges the response of the trained neural networks has no relation to the sin(x).

Although neural networks are inherently nonlinear, they can be trained to learn linear functions within a range. This is shown in Figure 2 where the training data is selected from a linear function within a range. It is clear that the trained neural network has learned to approximate a linear function within that range, while outside that range it exhibits a nonlinear response.

Neural networks can be trained and retrained. Retraining is an important capability of neural networks. When new data becomes available with enhanced information content, the new data can be added to the existing data set and used in retraining the neural network. In this case, the neural network will be learning the new information in the additional data. If the new data does not have any new information, the connection weights of the previously trained neural network do not change. It is not often possible to determine whether the new data has new information. This can be easily verified by monitoring the changes

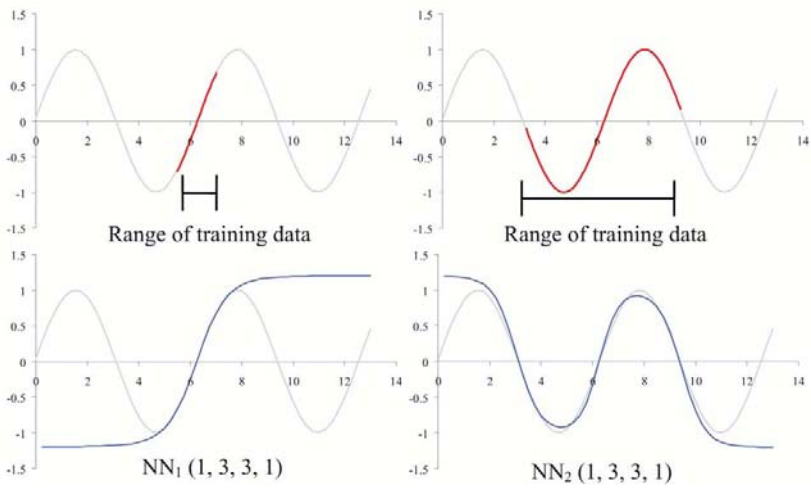


Figure 1. Neural networks trained to approximate sin(x) within two different ranges of x

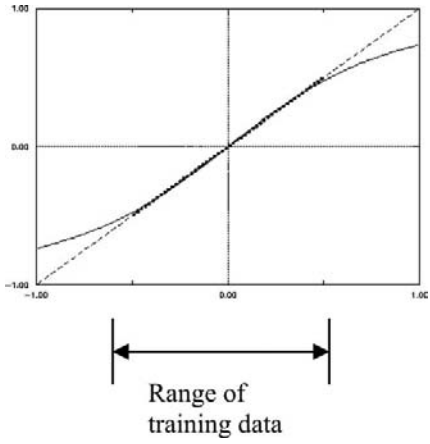


Figure 2. Neural network learns to approximate a linear function within two ranges of x.

in the connection weights during the retraining. If the neural network is retrained only with the new data, it will "forget" the information that was contained in the

data used in the original training. This is demonstrated in Figure 3. The upper part of the figure shows the range of the training data and the lower part of the figure shows the performance of the trained neural network.

The neural network NN(1, 3, 3, 1) was first trained with data from a range of x in sin(x) shown on the left. The same neural network was next retrained with data from another portion of the same function, shown on the right. It can be seen that the retrained neural network "forgot" the earlier information and learned to approximate the data within the new range.

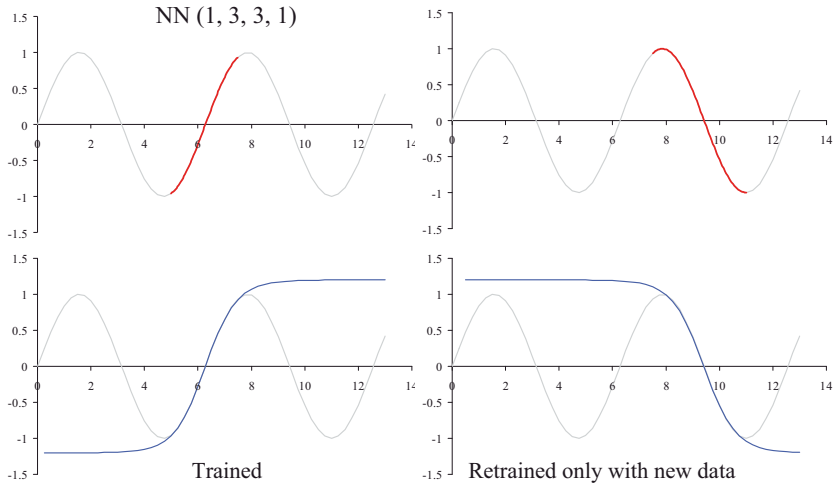


Figure 3. Retraining a neural network with new data.

Imprecision tolerance in neural networks creates a capability for dealing with scatter in data. The information in the training data sets for neural networks can contain many forms of information. Some information in the data is fairly dominant and easily stands out. Other types of information may appear weak and difficult to learn. All types of information may contain scatter, which itself can be considered a form of information. Neural networks can deal with scatter in data if trained properly. The danger in this case is overtraining that may force the neural network to learn the scatter as well. We will demonstrate these issues with a simple example.

Figure 4 shows a data set in the x, y plane. We wish to train a neural network to relate x (input) to y (output). Clearly, this data has a dominant pattern consisting of two nearly linear segments and a transition between them around $x = 8$.

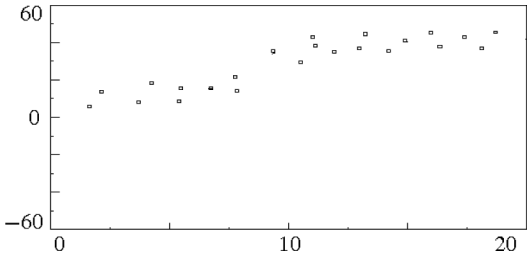


Figure 4. Data with scatter.

In the upper left part of Figure 5, the response of small neural network NN(1, 2, 2, 1) trained with the data with scatter is shown. It is clear that the neural network has learned the dominant pattern in the data well. If the training in the same neural network is continued and new nodes are added to the hidden layers adaptively (adaptive node generation will be discussed later), the response of the neural network is shown in the other parts of Figure 5. For example, in the upper right part of Figure 5 the neural network started with the same architecture as in the upper left, with 2 nodes in each of the two hidden layers, and during the continued training additional nodes were added to the hidden layer, ending with 5 nodes in each hidden layer. It can be seen that the neural network has started learning the scatter in the data. The process of training and adding new nodes to the hidden layers continued and two instances are shown in the lower part of Figure 5, when hidden layers reach 10 and 20 nodes each. The continued process of

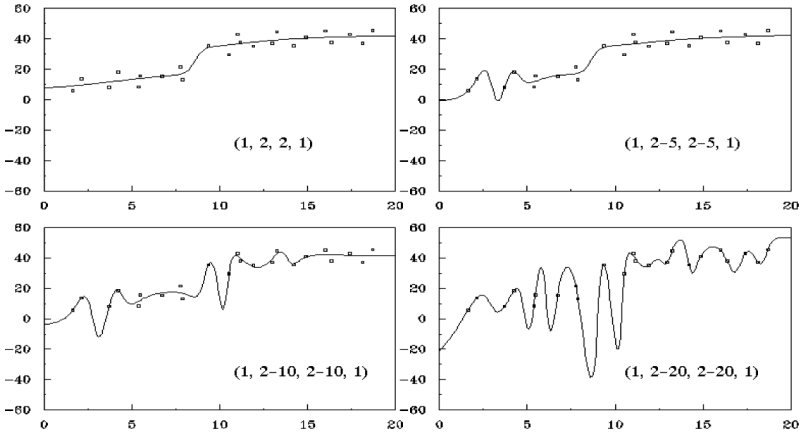


Figure 5. Training and over-training of a neural network with data containing scatter.

over-training is clearly visible. The part of Figure 5 on lower right shows a large neural network that is clearly over-trained and has learned to represent all the data points precisely. Although the over-trained neural network matches the data points, it may give seriously erroneous and unreasonable results between the data points.

It is important to be aware of the possibility of over-training the neural networks when dealing with measured and real data that often contain scatter. When dealing with numerically generated data, often the data does not contain any scatter. In those cases the over-training may not be obvious in the performance of the neural network, as is the case when there is scatter in the data. In some cases the nature of the information contained in the numerically generated data, and the distribution of the data point may also lead to similar phenomena as overtraining shown in Figure 5.

Transition from mathematical models to information contained in data . In working with neural networks it is always important to remember that the information that the neural network learns is contained in the data. The principle of functional non-uniqueness, discussed earlier, means that different neural networks can be trained to learn the information contained in the same data with some imprecision. In many applications the data changes and the information contained in the data also changes and is updated. We will see an example of this in the autoprogressive algorithm that will be discussed later in the chapter. At any point in the evolution of the data, it is possible to discard the old trained neural network and start with a new neural network and train it on the expanded and modified data set.

This observation leads to an important and profound point about the use of neural networks in modeling physical phenomena such as modeling of material behavior. In the mathematically based approaches, we use mathematical functions to model the physical phenomena. The information about the response of the physical system to the external stimuli is contained in that mathematical model.

When we use neural networks to model the physical phenomena, the information about the response of the physical system to the external stimuli is learned by the neural network from its training data. So, if we start thinking in terms of information - rather than mathematical models - then the neural networks are simply tools for extracting the information from the data and storing it. It is the learning capabilities of the neural networks that allow us to extract the information. Since we can extract the information from data at many stages, and with many different neural networks, it is the data and the information contained in that data that is the soft computing equivalent of the mathematical model and the information contained in the mathematical model. In moving from mathematical modeling to soft computing with neural networks, it is important that we start transitioning our thinking from mathematical model to the information contained in data.

6 Neural Networks in Material Modeling

Constitutive modeling is an inverse problem. In a typical material experiment stresses are applied to a specimen and the resulting strains are measured. If the material constitutive model is considered as a system, then the stresses and strains are the input and the output of the system. The material constitutive modeling is therefore an inverse problem, where the input and output are known and the system needs to be determined.

The conventional approach to this problem is to develop a mathematical model. Plasticity and other material modeling methods use mathematical modeling to represent the information present in the measured stresses and strains from the experiments. Neural networks can be used directly to extract the information from the experimental data. First application of this approach to model the constitutive behavior of plane concrete in two dimensions is reported in (Ghaboussi, Garrett and Wu 1990, 1991). Experimental results were used to directly train the neural network. It was shown that the neural networks can learn to relate the stresses to strains or vice versa. For practical use in finite element analysis it is preferable that the strains be used as the input to the neural networks and the stresses be used as the output. A typical neural network for a two dimensional material model can have the three components of strain as input and the three components of stress as output. However, this type of neural network is not suitable for representing the path dependence in constitutive behavior of materials.

Information required to represent the path dependence of material behavior consists of two parts: the current state of the material and immediate past history. The current state may consist only of the current state of stresses and strains, and this may be sufficient in absence of strong path dependence. In that case the input to the neural network may consist of the current state and the strains at the end of the increment (or the strain increments) and the output representing the stresses at the end of the increment (or the stress increments). As a simple example, we can consider a one-dimensional material behavior. At any point, given the values of the strain and the stress, the material behavior depends on whether the strains are increasing (loading) or decreasing (unloading). The current state and strain increments as input contain sufficient information to uniquely determine the stress increment as output.

Path dependence in multi-dimensions is far more complex and often the current state as input is not sufficient. Information about the past history of the state of strains and stresses is also needed. A three-point model was proposed in paper by Ghaboussi, Garrett and Wu, 1991 in which the input consisted of the strains and stresses at t_n (current state), and at t_{n-1}, t_{n-2} (history points) and the strain increments (or strains at t_{n+1}). The output of the neural network represented the stresses at t_{n+1}. This three point model was successfully applied to a number of materials.

Neural network were successfully used in the constitutive modeling of concrete, composites and geomaterials (see Ghaboussi, Garrett and Wu 1990, 1991; Wu 1991; Ghaboussi 1992a, 1992b; Wu and Ghaboussi 1993; Ghaboussi, Lade and Sidarta 1994; Ghaboussi and Sidarta, 1998; Sidarta and Ghaboussi, 1998, Ghaboussi, Pecknold, Zhang and HajAli 1998), strain softening material models in reinforced concrete (Kaklauskus and Ghaboussi, 2000), rate dependent material behavior (Jung and Ghaboussi, 2006a, 2006b), and hysteretic behavior of materials (Yun 2006, Yun Ghaboussi and Elnashai 2008a, 2008b, 2008c).

6.1 Nested Adaptive Neural Networks (NANN)

Often, it is difficult to determine the number of nodes in the hidden layers in a given problem. It was mentioned earlier that the size of the neural network is related to the complexity of the information contained in the training data. At present it is not possible to quantify the degree of complexity in the data. A method for adaptive node generation in neural networks was proposed in an article by Joghataie, Ghaboussi and Wu (1995). This method allows the training to start with a neural network with few hidden layer nodes and adaptively add new nodes to the hidden layers during the training, ending with a neural network that has the necessary architecture to learn the information in the data. This process is shown in Figure 6.

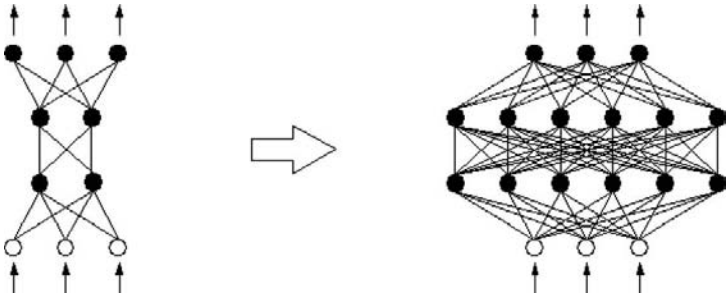


Figure 6. Adaptive node generation.

There are several steps in the adaptive node generation, illustrated in Figure 7. During the training the learning rate is monitored. The learning rate that is a measure of the change in the output errors normally reduces during the training. When the learning rate falls below a prescribed value, it is an indication that the capacity of the neural network is reached. At this point one node is added to each of the hidden layers. This creates a number of new connections that are assigned random initial weights. The old connection weights are frozen for a

number of epochs when the training resumes and only the new connection weights are modified. This is based on the rationale that the new connection weights are trained to learn the portion of the information that was not learned by the old connection weights. After a number of epochs (either prescribed, or adaptively determined by monitoring the learning rate) the old connections are unfrozen and the training continues.

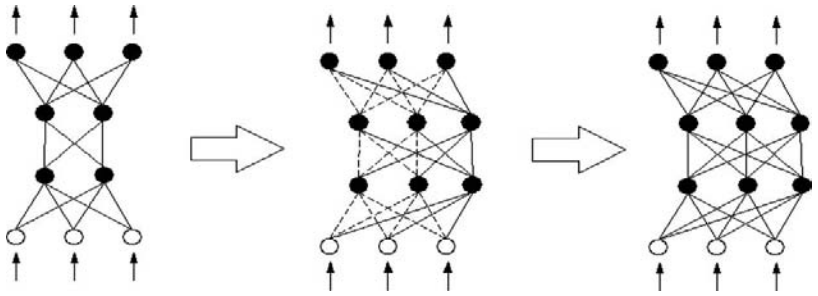


Figure 7, Process of adaptive node generation.

The nested part of "nested adaptive neural networks" deals with the structure of the information in the training data. In most cases the information in the training data can have a clear structure. This structure may not be obvious from the data itself but it can be discerned from the process that was used to generate the training data. In some situations the structure in the training data can be exploited and used in the design of the neural network architecture, or in designing a collection of neural networks.

An example of the structure of the information in the training data is the nested structure; it is possible to recognize subsets of the data that have a clear hierarchical nested structure. Starting from the lowest subset, each data subset is in turn a subset of the next higher level data subset (see Ghaboussi Zhang Wu and Pecknold 1997, Ghaboussi and Sidarta, 1998).

Consider the example of the training data in constitutive behavior of the materials. In this case, one type of data hierarchy comes from the dimensionality. The data from one-dimensional constitutive material behavior is a subset of the data from two-dimensional constitutive material behavior that in turn is a subset of the data from three dimensional material behavior. The same nested structure is present in the information contained in the training data. If we assume that the functions f_j describing the material behavior in 1D, 2D, axisymmetric, and 3D, belong to 1, 3, 4, and, 6 dimensional function spaces F_j.

$$\Phi_j = f_j(f_j) \qquad j = 1, 3, 4, 6$$

$$\boldsymbol{f}_j \; 0 \; \boldsymbol{F}_j$$

Then, these function spaces have a nested structure.

$$\boldsymbol{F}_1 \; \delta \; \boldsymbol{F}_3 \; \delta \; \boldsymbol{F}_4 \; \delta \; \boldsymbol{F}_6$$

A similar type of nested structure is present in the information in the data generated from time-independent and time-dependent material behavior. In this case, the information on the time-independent material behavior is a subset of the information on the time-dependent material behavior.

The nested structure of the information in the data can be reflected in the structure of the neural networks. As many neural networks as the subsets can be developed with the hierarchical relationship between those neural networks. A base module neural network can be trained with the lowest level subset, in this case the one-dimensional material behavior. Next a second level module can be added to the base module that represents the information on the two-dimensional material behavior that is not represented in the base module. The connections between the second level module and the base module have to be one way connections; there are connections from the second level module to the base module, but no connections from the base module to the second level module. The information on one-dimensional material behavior does not contribute to the additional information needed for the two-dimensional behavior. This is the reason behind the one way connection. Similar hierarchies will be present when the third and fourth level modules are added to represent the axisymmetric and the three-dimensional material behavior.

Another example of the nested structure of information in the data is in modeling the path dependency of material behavior with history points. In the first application of neural networks in constitutive modeling (see Ghaboussi, Garrett and Wu, 1991) path dependency in loading and unloading was modeled with three history points. When path dependency is not important, it is sufficient to provide the current state of stresses and strains and the strains at the end of the increment as input to obtain the state of stresses at the end of the increment. This information is not sufficient when path dependency becomes important, for instance, when unloading and reloading occurs. In this case, the material behavior is influenced by the past history. In this case path dependency was modeled by additionally providing the state of stresses and strains at three history points. This was called the "three point model."

The following equation shows the path dependent material behavior modeled with history points. Function f_{jk} relates the stress rate to strain rate, where subscript j represents the dimension and subscript k represents the number of history points.

$$\Phi_j = f_{jk}\,(\Phi_{j,0},\ \varepsilon_{j,0},\ \Phi_{j,-1},\ \varepsilon_{j,-1},\ \ldots,\ \Phi_{j,-k},\ \varepsilon_{j,-k},\ f_j)$$

$$j = 1, 3, 4, 6\ ;\ k = 0, 1, 2, \ldots$$

Functions f_{jk} belong to function spaces F_{jk} that have a nested relationship.

$$f_{j,k}\ 0\ F_{j,k}$$

A nested adaptive neural network for a three point model of a one-dimensional material behavior is illustrated in Figure 8. Figure 8a shows the adaptive training of the base module. The input to the base module consists of the current state of stresses and strains and the strain increment. Figures 8b and 8c show the addition of the second and third history points and their adaptive training. Each history module has only one-way connections to the previous modules. The reason for the one-way connections is obvious in this case; the past can influence the present, but the present has no influence on the past.

Nested adaptive neural networks were first applied to an example of one-dimensional cyclic behavior of plain concrete (see Zhang, 1996). NANNs were trained with the results of one experiment and the trained neural network was tested with the results of another experiment performed at a different laboratory. First, the base module was trained adaptively. As shown in the following equation, the input of the neural network is strain and the output is stress increment. Each of the two hidden layers started the adaptive training with 2 nodes and ended with four nodes.

$$\in\Phi_{n+1} = NN_0\ [\varepsilon_{n+1}\colon 1, 2-4, 2-4, 1]$$

The notation in this equation was introduced to describe both the representation and the network architecture. The left hand side of the equation shows the output of the neural network. The first part inside the square brackets describes the input to the neural network and the second part describes the network architecture.

Next, the first history point module was added and the resulting neural network is shown in the following equation. The first history module had two input nodes, no output node, and each of the two hidden layers started the adaptive training with 2 nodes and ended with 12 nodes.

$$\in\Phi_{n+1} = NN_1\ [\varepsilon_n,\ \Phi_n,\ \varepsilon_{n+1}\colon (1, 2), (2-4, 2-12), (2-4, 2-12), (1)]$$

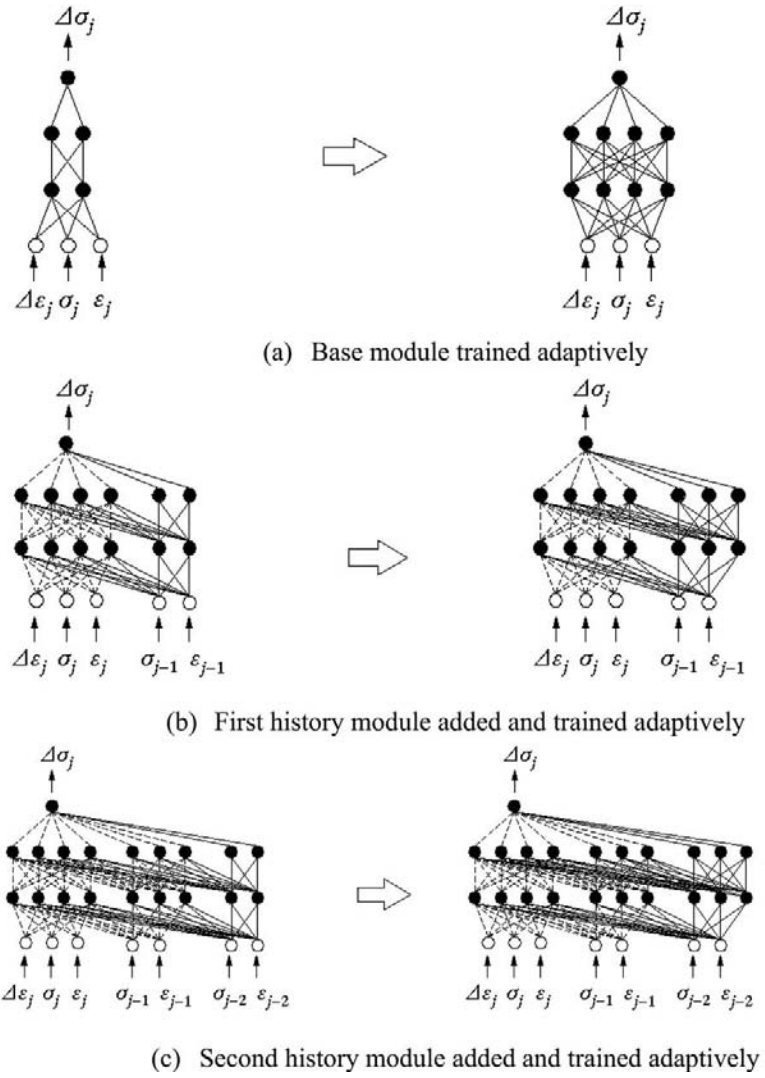


Figure 8. Nested adaptive neural network for material behavior in 1D

The performance of the neural network NN_1 on the training and testing experimental results is shown in Figure 9a.

The following equation shows the NANN after the second history point was added and adaptively trained.

$$\in\Phi_{n+1} = NN_2\ [,_n, \Phi_n, ,_{n-1}, \Phi_{n-1}, \ ,_{n+1}: (1, 2, 2),$$
$$(2–4, 2–12, 2–10), (2 –4, 2–12, 2–10), (1)]$$

In this case, the hidden layers in the second history module started the adaptive training with 2 nodes and ended with 10 nodes. The performance of the NANN NN_2 with two history points is shown in Figure 9b.

The following equation shows the NANN after the third history point was added and adaptively trained.

$$\in\Phi_{n+1} = NN_3\ [,_n, \Phi_n, ,_{n-1}, \Phi_{n-1}, ,_{n-2}, \Phi_{n-2}, ,_{n+1}: (1, 2, 2, 2),$$
$$(2–4, 2–12, 2–10, 2–9),$$
$$(2 –4, 2–12, 2–10, 2–9), (1)]$$

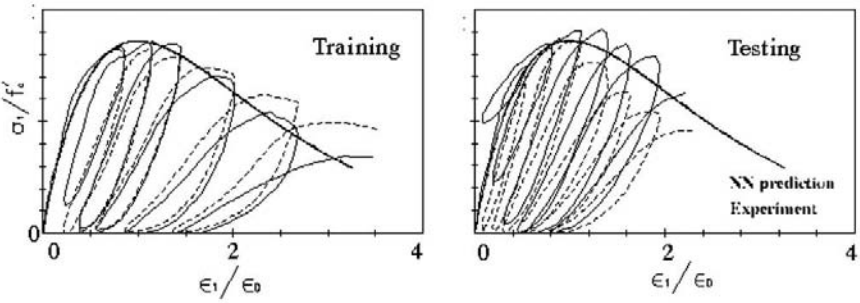
In this case, each of the two hidden layers in the third history module started the adaptive training with 2 nodes and ended with 9 nodes. The performance of the NANN NN_3 with three history points is shown in Figure 9c.

The cyclic behavior of plain concrete shown in Figures 9a, 9b and 9c clearly exhibits a strong path and history dependence. In order to capture this history dependent material behavior, history points are needed and this is clearly shown in Figures 9a, 9b and 9c by the fact that the performance of the NANN improves with the addition of the new history points. The neural network NN_3 with three history points appears to have learned the cyclic material behavior with a reasonable level of accuracy. These figures also demonstrate that the trained neural network has learned the underlying material behavior, not just a specific curve; the trained neural network is able to predict the results of a different experiment that was not included in the training data.

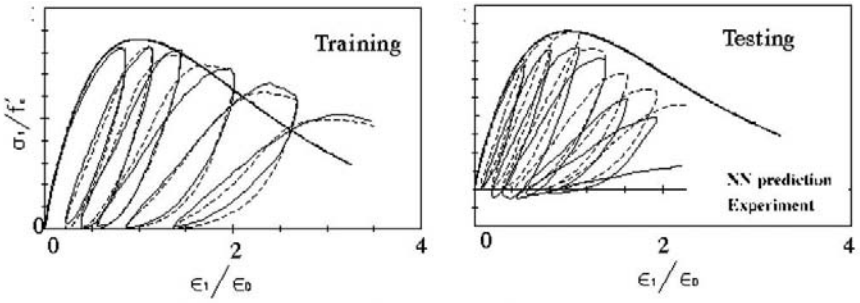
6.2 Neural Networks for Hysteretic Behavior of Materials

In the previous section it was demonstrated that by using the history points, such as the three point model, it is possible to model the hysteretic behavior of materials. In this section we will describe a new method for modeling the hysteretic behavior of materials (see, Yun, 2006, Yun, Ghaboussi, Elnashai 2008a).

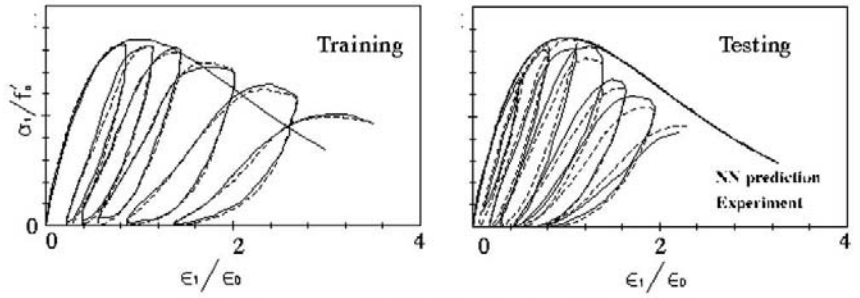
If we look at a typical stress-strain cycle, it is easy to see that the value of strain is not sufficient to uniquely define the stress, and vice versa. We refer to this as one-to-many mappings. Neural networks can not learn one-to-many mappings. This can be remedied by introducing new variables at the input to



(a) Performance of NANN with one history point.



(b) Performance of NANN with two history points.



(c) Performance of NANN with three history points

Figure 9. Performance of NANN with one, two and three history points (Zhang, 1996).

create a unique mapping between the input and output to the neural network. The variable used in this case is given in the following equation:

$$O_n = \Phi^T{}_{n-1}, n$$

This parameter strictly guarantees one-to-one mapping only in one-dimension. However, our experience shows that it is also effective in multi-dimension. The variable 0_n is the sum of two quantities. The two parts of the hysteretic variable are described in the following equations and illustrated in Figure 10.

$$0_n = >_n + \in 0_n$$

$$>_n = \Phi^T{}_{n-1,n-1}$$

$$\in 0_n = \Phi^I{}_{n-1} \in_{,n}$$

The first variable $>_n$ uniquely helps locate the point on the stress-strain diagram or the current state of stresses and strains. The second variable $\in 0_n$ indicates the direction of the increment of strains.

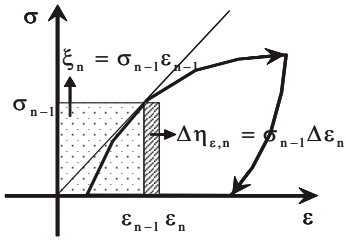


Figure 10. Hysteretic variables on one-dimensional stress-strain diagram.

These two variables have been found to be effective in the modeling of the hysteretic behavior of materials. They can be used on their own, or in combination with one or two history points. A typical neural network material model capable of learning the hysteretic behavior of materials is shown in Figure 11 and its application is shown in Figures 12 and 13. The input to the neural network consists of the current stresses and strains, hysteretic variables, and the strains at the end of the increment. The output is the stresses at the end of the increment.

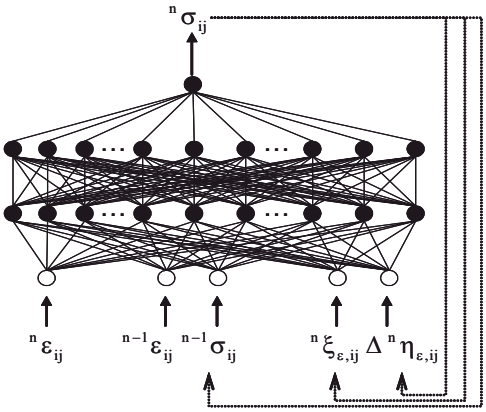


Figure 11. Neural network for hysteretic behavior of materials [Yun, 2006]

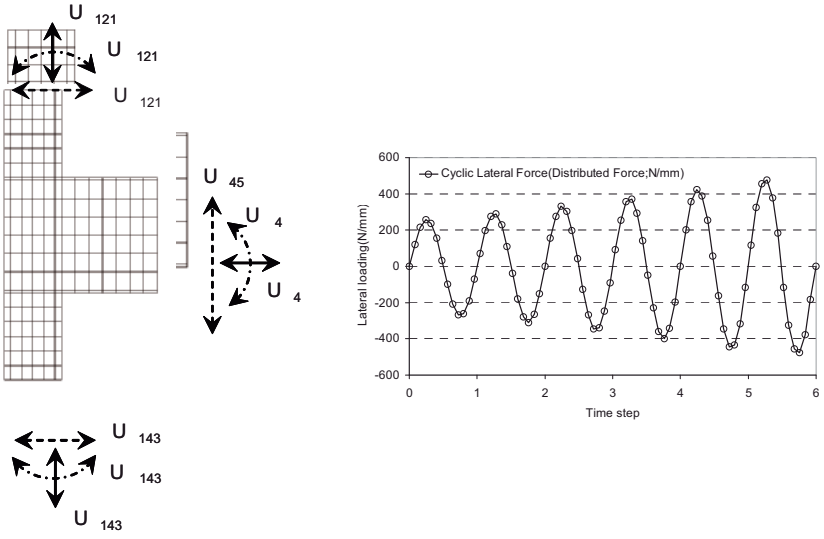


Figure 12. FE model of beam column connection subjected to cyclic force [Yun 2006].

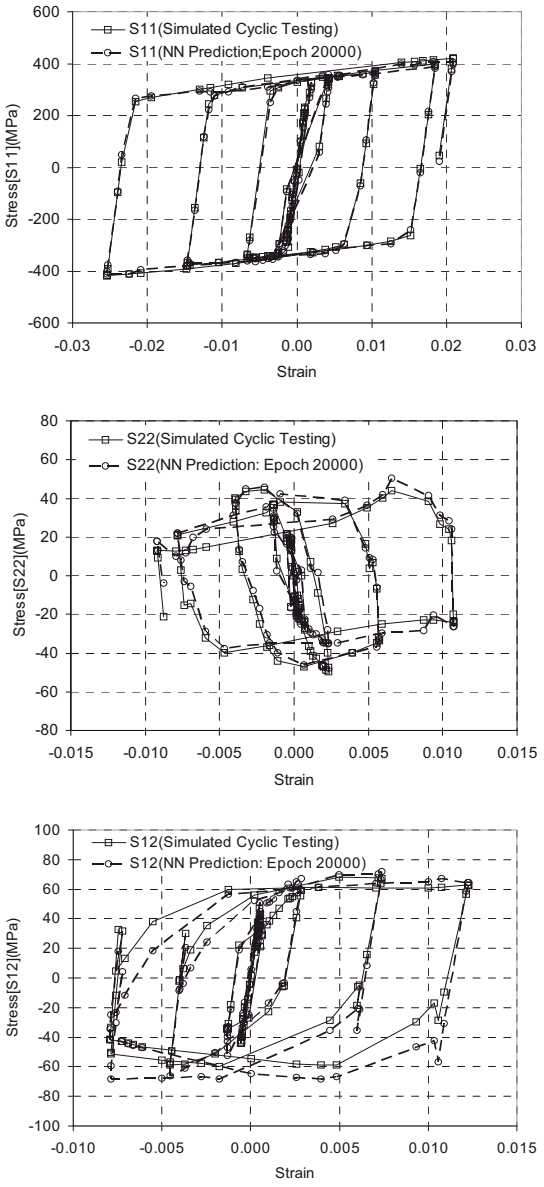


Figure 13. Performance of the training neural network material model with hysteretic variables (Yun 2006, Yun, Ghaboussi, Elnashai, 2008a).

6.3 Autoprogressive Algorithm and Self-learning Simulations (SelfSim)

In the early applications of neural networks in constitutive modeling they were trained directly with the data from material tests. Constitutive modeling from the material tests is an inverse problem. In this case the input (applied stresses) and the output (measured strains) are known and the system that in this case is the constitutive model needs to be determined. In the conventional mathematically based methods this is done by developing a mathematical model that closely matches the observed behavior. Of course, the mathematical models have to conform to the conservation laws of mechanics. As we have discussed before, in using neural networks we concentrate on the information on the material behavior contained in the data generated from the material tests. The learning capabilities of neural networks provide the means for extracting and storing the information on material behavior directly from data generated by the material tests.

The advantage of extracting the information on the material behavior directly from the experimental results is that there is no need for idealization. Moreover, since the data comes from the material itself, it is safe to assume that it conforms to the conservation laws of mechanics. There are also disadvantages that became clear very soon after the initial applications. Material tests are designed to represent a material point. Consequently, the state of stresses and strains within the sample must be as uniform as possible. The sample is subjected to a stress path and all the points within the sample are assumed to follow the same stress path. Therefore, the data generated from a material test has information about the material behavior only along that stress path. The data from one material test is not sufficient to train a robust neural network material model. Information on the material behavior over the whole region of interest in stress space is needed to train a robust neural network constitutive model with generalization capabilities. This requires a series of specially designed material tests with stress paths that reasonably cover the region of interest in the stress space. This is not practical in most cases.

Material tests are not the only source of information on the behavior. There are many other potential sources of data that contain information on material behavior. The measured response of a system subjected to a known excitation contains information on the constitutive behavior of the material (or materials) in that system. An example is a structural test; the measured displacements of a structure that is subjected to known forces contain information on the constitutive properties of the materials within the structure. This is a more complex inverse problem than the material tests; the forces and displacement are the known input and output of the system and the constitutive properties of the material within the structure are to be determined. Unlike material tests, which ideally induce a uniform state of stress within the sample, structural tests induce non-uniform states of stresses and strains within the sample. Since points in the

specimen follow different stress paths, a single structural test potentially has far more information on the material behavior than a material test in which all the points follow the same stress path.

Extracting the information on the material properties from structural tests is an extremely difficult problem with the conventional mathematically based methods. This is probably the reason why it has not been attempted in the past. On the other hand, soft computing methods are ideally suited for this type of difficult inverse problem. The learning capabilities of a neural network offer a solution to this problem. Autoprogressive algorithm is a method for training a neural network to learn the constitutive properties of materials from structural tests(see Ghaboussi, Pecknold, Zhang and HajAli,1998).

An autoprogressive algorithm is applied to a finite element model of the structure being tested. Initially a neural network material model is pre-trained with an idealized material behavior. Usually, linearly elastic material properties are used as the initial material model. Sometimes it may be possible to use a previously trained neural network as initial material model. The pre-trained or previously trained neural network is then used to represent the material behavior in a finite element model of the specimen in the structural test. The autoprogressive method simulates the structural test through a series of load steps. Two analyses are performed in each load step. The first analysis is a standard finite element analysis, where the load increment is applied and the displacements are computed. In the second analysis the measured displacements are imposed. The stresses from the first analysis and the strains from the second analysis are used to continue the training of the neural networks material model. The stresses in the first analysis are closer to the true stresses since the first analysis is approaching the condition of satisfying the equilibrium. Similarly, the strains in the second analysis are closer to the true strains since it is approaching the condition of satisfying compatibility. A number of iterations may be required in each load step. The retrained neural network is then used as the material model in the next load step. The process is repeated for all the load steps and this is called one “pass”. Several passes may be needed for the neural network to learn the material behavior. At that point, the two finite element analyses with the trained neural network will produce the same results.

6.4 An Illustrative Example: A Truss Arch

The autoprogessive algorithm is illustrated with a three-dimensional truss example, shown in Figure 14. This structure represents the left half of a symmetric truss arch. The nodes at the right hand side have symmetric boundary conditions and the nodes at the left hand side are fixed, representing the fixed support condition.

It is assumed that the members of the truss are made from a hypothetical material with nonlinear stress-strain relations shown in Figure 15. The objective

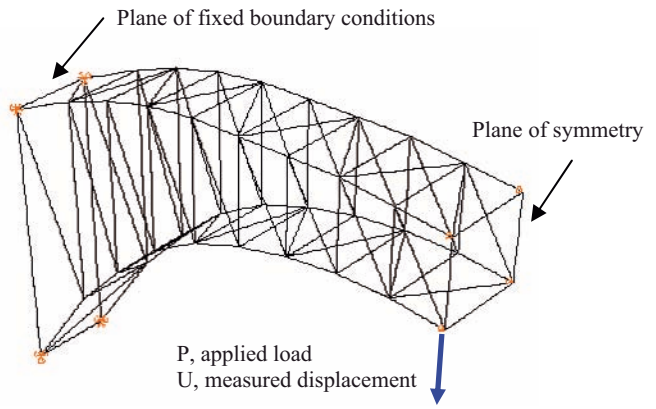


Figure 14. Three-dimensional truss example

of autoprogressive algorithm in this example is to train a neural network to learn the nonlinear stress-strain relation shown in Figure 15 from the result of a structural test. In this case the structural test consists of applying a single force P at one node and measuring the displacement at the same point along the direction of the applied load, as shown in Figure 14. The experiment is simulated by analyzing the structure. The force-displacement curve from the simulated structural test is shown in Figure 16.

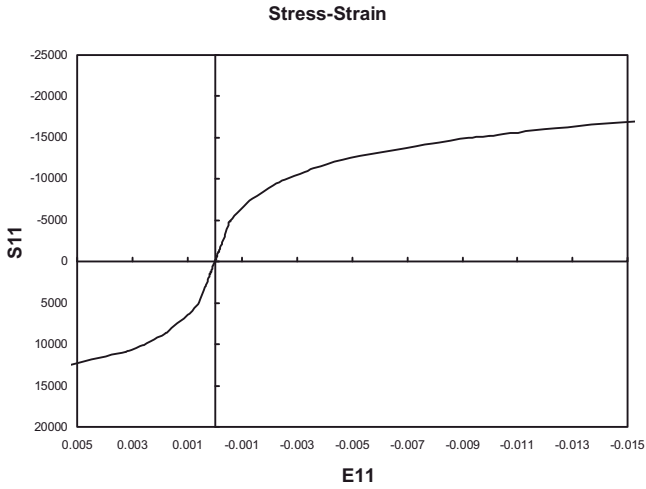


Figure 15. Assumed member stress-strain relation

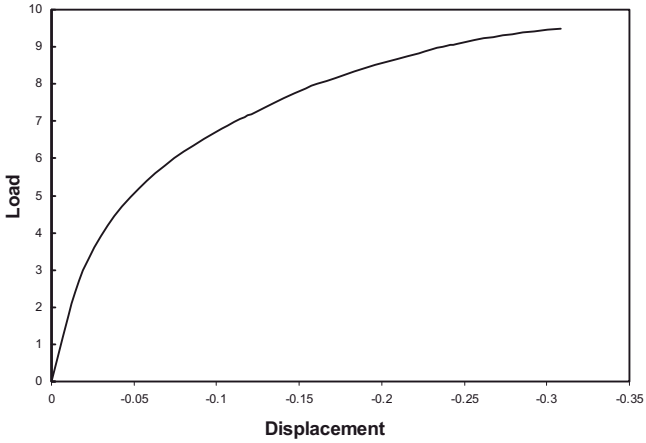


Figure 16. Force-displacement curve from simulated structural test.

Initially, the neural network is trained with linearly elastic behavior within a narrow range of stresses and strains. This neural network is used in the self-learning simulation. The load P and the displacement U are applied in increments in the first and second finite element analyses (FEM-A and FEM-B) and the results are used to retrain the neural network material model. A total of 10 passes are performed. At several stages during the self-learning simulation, the partially trained neural network is used in a forward analysis simulation of the structural test. Force-displacement relations at six stages during the self-learning simulation are shown in Figure 17. Show in Figure 18 is the process of the gradual learning of the material behavior. Points in this figure represent the state of stress and strain in the structural members. After 10 passes the neural network has learned the material behavior so that the forward analysis with the trained neural network replicates the simulated structural test with sufficient accuracy.

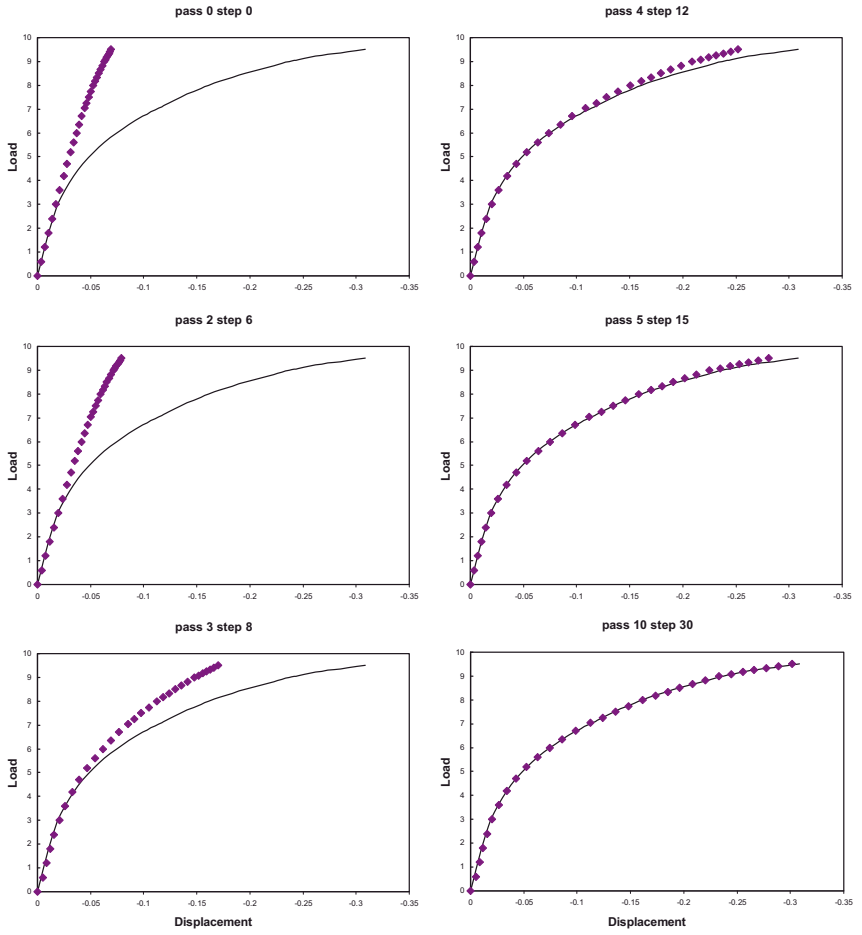


Figure 17. Force-displacement relations in the forward analyses with the neural network material model at six stages during the autoprogressive training in the self-learning simulation

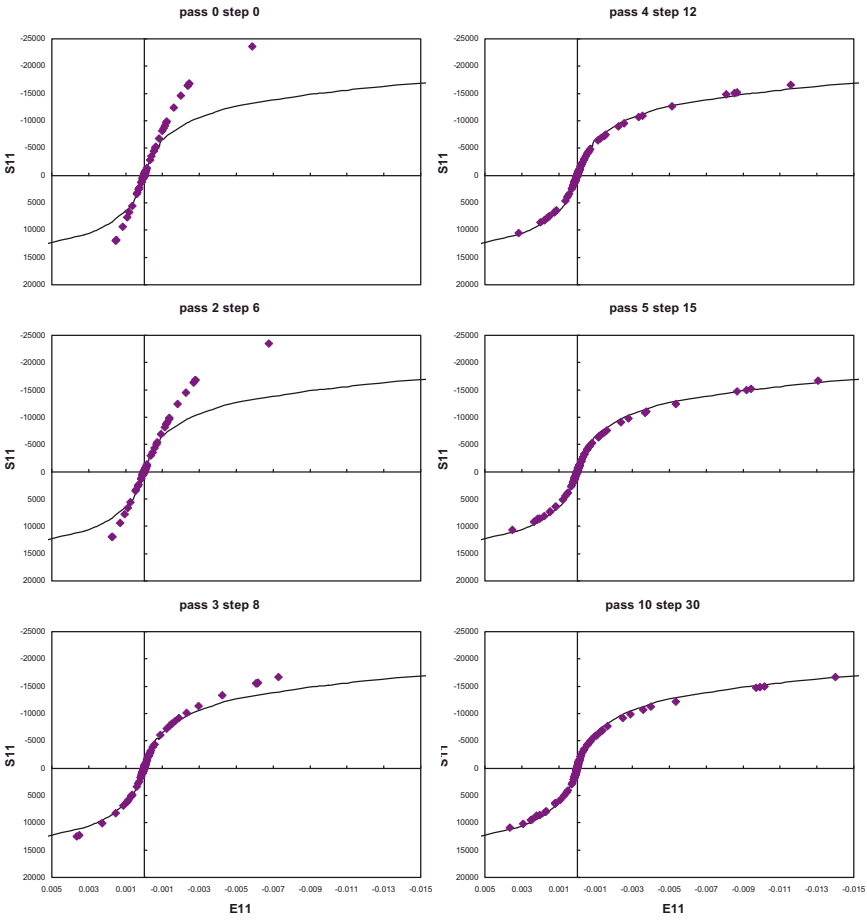


Figure 18. Stress-strain relations in the forward analyses with the neural network material model at six stages during the autoprogressive training in the self-learning simulation

6.5 Autoprogressive Algorithm and Self-learning Simulations in Structural Mechanics and in Geo-mechanics

The problem of determining the constitutive properties of material from structural tests is more relevant with modern composite material. It is often not feasible to perform comprehensive sets of material tests on composite materials. However, structural tests on composite materials are far more common. An

application of the Autoprogressive method to a composite plate with a hole is presented in paper by Ghaboussi Pecknold Zhang and HajAli (1998).

The Autoprogressive algorithm has been successfully applied in self-learning simulations in deep excavations in urban areas (see Hashash Marulando Ghaboussi and Jung 2003, 2006), in modeling of the behavior of geo-materials from non-uniform tests (see Sidarta and Ghaboussi, 1998; Fu Hashash Jung and Ghaboussi, 2007; Hashash Fu Ghaboussi Lade and Saucier, 2009), in time dependent behavior of concrete in segmented long span bridges (see Jung Ghaboussi and Marulando, 2007), and in modeling of hysteretic behavior of beam-column connections from the results of dynamic tests on frames (see Yun Ghaboussi and Elnashai 2008b, 2008c).

6.6 Self-learning Simulations in Bio-medicine

Another broad area of application self-learning simulation with autoprogressive algorithm is in bio-medicine and in bio-medical imaging. Direct in-vivo experiments to determine constitutive properties of soft tissue are not currently possible. However, minimally invasive in-vivo methods for measuring the system response of soft tissue are possible. Self-learning simulation has been successfully applied in determination of the constitutive properties of the human cornea (see Kwon, 2006; Kwon Ghaboussi Pecknold and Hashash 2008, 2009). The same method is also used in the accurate determination of Intra-ocular Pressure (IOP). Accurate determination of IOP and constitutive properties of the human cornea can be used in virtual laser surgery to individualize and optimize the laser surgery procedure.

6.7 Self-learning Simulations in Complex Systems

The autoprogressive method has wider applications than material modeling from structural tests. System response is dependent on the properties of its components. In many cases it is not possible to determine the properties of the components of the systems through experiments; such experiments many not be possible. However, in most cases it is possible to determine the input/output response of the system. The measured input and output of any system can be used to develop a neural network model of the behavior of its components. The requirement for the application of autoprogessive algorithm is that a numerical modeling of the system be feasible so that the simulation of system response to the input can be performed in the dual analyses similar to the self-learning simulation in structural tests. The potential application of the autoprogressive algorithm in characterization of the components of complex systems is described in paper by Ghaboussi, Pecknold and Zhang, 1998.

7. Inverse Problems in Engineering

Most engineering problems are inherently inverse problems. However, they are seldom solved as inverse problems. They are only treated as inverse problems on the rare occasions when they can be formulated as direct problems in highly idealized and simplified forms. Nearly all the computer simulations with hard computing methods, including finite element analyses, are direct or forward problems, where a physical phenomenon is modeled and the response of the system is computed. There are two classes of inverse problems. In one class of inverse problems the model of the system and its output are known. The inverse problem is then formulated to compute the input which produced the known output. In the second class of inverse problems the input and the output of the system are known, and the inverse problem is formulated to determine the system model. This class of inverse problems is also referred to as system identification.

An important characteristic of the inverse problems is that they often do not have mathematically unique solutions. The measured output (response) of the system may not contain sufficient information to uniquely determine the input to the system. Similarly, the measured input and response of the system may not contain enough information to uniquely identify the system, and there may be many solutions which satisfy the problem.

Lack of unique solutions is one of the reasons why the mathematically based methods are not suitable for inverse problems. On the other hand, biological systems have evolved highly robust methods for solving the difficult inverse problems encountered in nature. In fact, most of the computational problems in biological systems are in the form of inverse problems. The survival of the higher level organisms in the nature depends on their ability to solve these inverse problems. The examples of these inverse problems are recognition of their food, recognition of threats, and paths for their movements. Nature's basic strategy for solving the inverse problems is to use imprecision tolerant learning and reduction in disorder within a domain of interest.

A training data set can be used to train a forward neural network or an inverse neural network. The input and output of the inverse neural network are the same as the output and input, respectively, of the direct neural network. This was demonstrated in the first application of neural networks in material modeling (see Ghaboussi, Garrett and Wu, 1991).

When unique inverse relationship does exist, then both neural networks and mathematically based methods can be used to model the inverse mapping. However, when the inverse mapping is not unique, modeling with the mathematically based methods becomes increasingly difficult, if not impossible. On the other hand, neural network based methods can deal with the non-unique inverse problems by using the learning capabilities of the neural networks. In fact, this is how the biological systems solve the inverse problems, through

learning. Even if mathematically unique inverse mappings do not exist, approximate (imprecision tolerant) inverse mappings may exist over limited regions of domain and range. Neural networks use learning to establish reasonable admissible inverse mappings that are valid over limited ranges of these variables.

In the previous sections the inverse problem of constitutive modeling was presented, including the autoprogressive algorithm in self-learning simulation. Other examples of application of soft computing methods to inverse problems in engineering from the work of the author and his co-workers will be briefly described in this section. More details on these applications are available in the references cited.

7.1 Inverse Problem of Structural Control

The objective of structural control is to minimize the motion of the structure through one or more actuators that can perform various functions such as apply forces to the structure or change the properties of some structural components. The controller receives input signals (such as measured motion of the structure) and sends a output signals to the actuators to accomplish the control objective. This is a task that can be performed by a trained neural network. The method for training of the neuro-controller is not often obvious. Depending on the feasibility of the type of data acquisition, the training of the neural controller can be accomplished in different ways. The neural network is learning the control method – in the context of inverse problems it is learning the system model from the input and output. However, in most cases the output that is the desired actuator signal is not known.

In the first application of neural networks in structural control (see Nikzad, Ghaboussi and Paul, 1996), it was possible in an experiment to send signals to the actuator and measure the system response. This data was used to train an inverse neural network so that the trained neural network could output a actuator signal to counter the response of the uncontrolled system.

It is not always possible to perform experiments to measure the structural response to actuator signals. In that case, the output of the neural controller is not known but the desired response of the structure is known. A new method for training of the neuro-controller was developed that involved the use of a pre-trained emulator neural network (see Ghaboussi and Joghataie, 1995; Bani-Hani and Ghaboussi, 1998a, 1998b; Bani-Hani Ghaboussi and Schneider, 1999a, 1999b).

The new method was verified in a number of experiments on control of structural response when the structure is subjected to earthquake ground motion. The results showed that the neural networks can be successfully used in structural control. Moreover, the neuro-controller can also learn to compensate for control system time-delays.

The inverse problem of structural control can also be solved by genetic algorithm (see Kim and Ghaboussi, 1999; Kim, 2000). In this case the controller evolves to satisfy the control criterion.

7.2 Inverse Problem of Generating Artificial Spectrum Compatible Accelerograms

Given an earthquake accelerogram, it is possible to compute its Fourier spectrum or response spectrum. This can be considered the forward problem. We consider the two inverse problems in this case; the input being either the Fourier spectrum or the response spectrum and the output being the earthquake accelerogram. Given a Fourier spectrum it is possible to uniquely determine the earthquake accelerogram that produced it. The reason for the uniqueness in this case is that no information is lost in the forward problem of going from the accelerogram to the Fourier spectrum. On the other hand, information is lost in the forward problem of going from accelerogram to response spectrum. Therefore, given a response spectrum, it is not possible to uniquely determine the accelerogram that produced it. This problem is of practical interest, since earthquake response spectra are used in design.

The inverse problem of generating artificial earthquake accelerograms from design response spectra is very similar to the inverse problem of recognizing faces or voices. Although unique solution does not exist, we learn to recognize a limited number of faces and voices. The same learning strategy can be used in generating artificial earthquake accelerograms. The author and his co-workers have developed a neural network based method for generating artificial spectrum compatible accelerograms (see Lin, 1999; Ghaboussi and Lin, 1998; Lin and Ghaboussi, 2001). The objective is to train a neural network with an ensemble of recorded earthquake accelerograms and their computed response spectra. The input to the neural network will be a vector of the discretised ordinates of the response spectrum. The output of the neural network is the vector of discretised ordinates of the real and imaginary parts of the Fourier spectrum. As mentioned earlier, the accelerogram can then be uniquely computed from the Fourier spectrum.

The neural networks are trained in two stages. First, a Replicator Neural Network (RNN) is trained to compress the information content of the Fourier spectra of the accelerograms in the training data sets. As shown in Figure 19 the input and output of the RNN are the same. The middle hidden layer of RNN has few nodes. The activations of the middle hidden layer nodes represent the compressed data. It is obvious that the RNN is performing two distinct functions of encoding and decoding in such a way that it can be separated into two neural networks. The lower part of RNN (input layer to the middle hidden layer) can be considered as an encoder neural network. Similarly the upper part of the

RNN (from the middle hidden layer to the output layer) can be considered as a decoder neural network.

The upper part of the trained RNN is used in the accelerogram Generator Neural Network (AGNN) shown in Figure 20. The lower part of the AGNN is then trained with a number of recorded earthquake accelerograms. The methodology has been extended to develop and train stochastic neural networks which are capable of generating multiple accelerograms from a single input design response spectrum (see Lin, 1999; Lin and Ghaboussi, 2001).

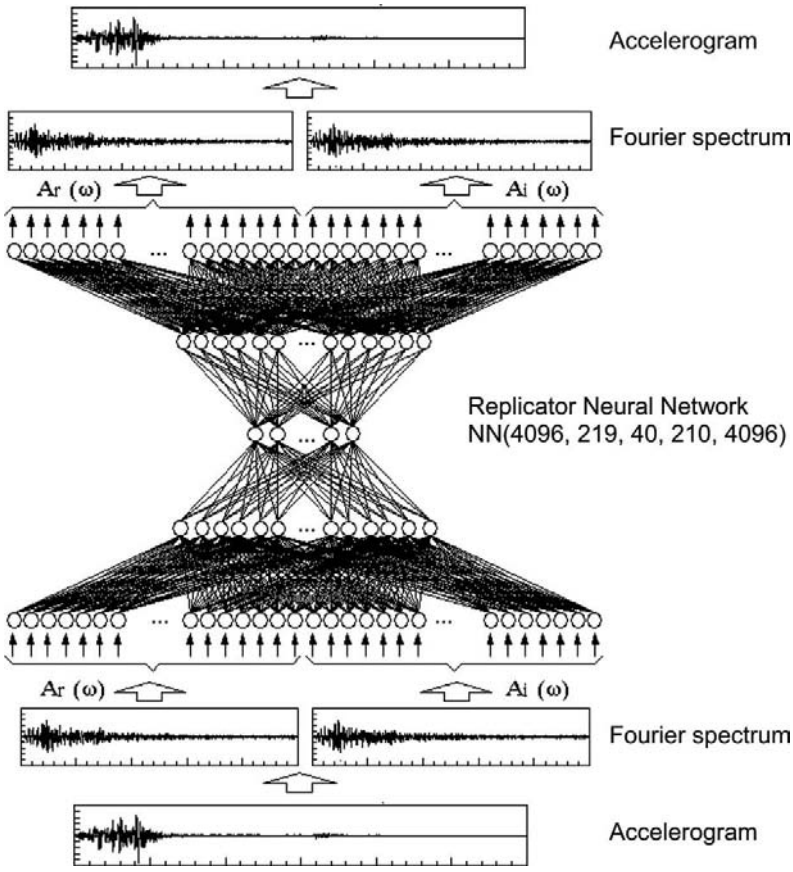


Figure 19. Replicator neural network

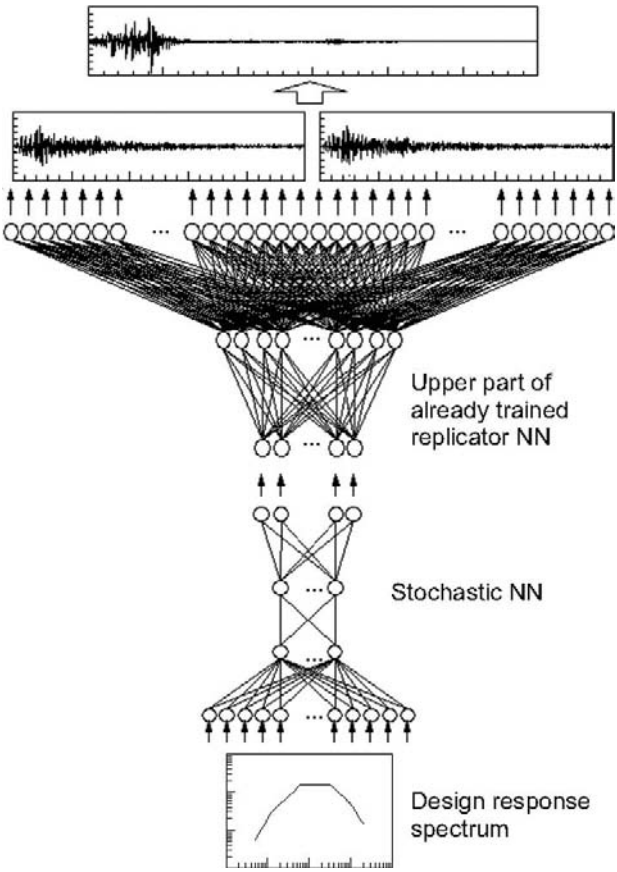


Figure 20. Composite Accelerogram Generator Neural Network

7.3 Inverse Problem of Condition monitoring and damage detection in bridges

Another obvious class of inverse problems is the condition monitoring and damage detection from the measured response of the structure. Soft computing methods again offer unique opportunities for dealing with this class of difficult problems. The first application of neural networks in structural condition monitoring was reported in [Wu Ghaboussi and Garrett, 1992]. It was shown that it is possible to train a neural network to learn the structural response when various states of damage, including no damage, are present in the structure. The trained neural network was shown to be capable of indicating the location and

extent of the damage from the structural response during an earthquake. Neural networks have also been applied in engineering diagnostics (see Ghaboussi and Banan, 1994) and in damage detection and inspection in railways (see Ghaboussi, Banan, and Florom, 1994; Chou, Ghaboussi, and Clark, 1998).

Application of genetic algorithm in bridge damage detection and condition monitoring was reported in papers by Chou (2000), Chou and Ghaboussi (1997), (1998), (2001). The bridge condition monitoring is formulated in the form of minimizing the error between the measured response and the response computed from a model of the bridge and in the process the condition of the bridge is determined. The methodology was shown capable of determining the condition of the bridge from the ambient response under normal traffic load. The methodology was further refined by Limsamphancharon (2003). A form of fiber optic sensing system was proposed for monitoring of the railway bridges under normal train traffic loads. A practical method was proposed for detecting the location and size of fatigue cracks in plate girders in railway bridges. A new Dynamic Neighborhood Method genetic algorithm (DNM) was proposed (see Limsamphancharon, 2003) that has the capability of simultaneously determining multiple solutions for a problem. DNM was combined with a powerful genetic algorithm called Implicit Redundant Representation GA (IRRGA) (see Raich and Ghaboussi, 1997a) which was used in bridge condition monitoring.

7.4 Inverse Problem of Creative Structural Design

Engineering design is probably the most fundamental engineering task. If we stay in the domain of structural engineering and consider the example of structural design, we can define the inverse problem as shown in Figure 21. The input is loads and design specifications, and the output is the code requirements, safety, serviceability and aesthetic. The objective is to determine a structural design that can carry the loads and meet the design specification while satisfying the output. If we pose this as an inverse problem, the methodology should directly seek a solution. Such a methodology currently does not exist. Instead, this problem is solved through a series of trial and errors; each trial constituting a forward problem.

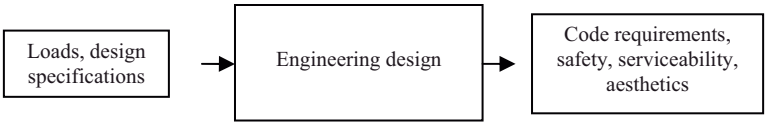


Figure 21. Inverse problem of structural design

Similar to the other engineering inverse problems, engineering design does not have a unique solution. A truly unconstrained inverse problem of engineering design is a search in an infinite dimensional vector space. There are

no optimum solutions and there are many solutions that will satisfy the input and output of the problem. As we constrain the problem, we also reduce the dimensions of the vector space where the search for the design takes place. A fully constrained design problem is reduced to an optimization problem that is a search in a finite dimensional vector space. Mathematically based hard computing methods have been used in optimization problems. It is far more difficult for the hard computing methods to deal with the unconstrained design problems.

As an example, we will consider the problem of designing a bridge to span a river. Soft computing methods can be used to solve this problem in a number of ways. The spectrum of all possible formulations is bound by the fully unconstrained design problem at one extreme and the fully constrained optimization problem at the other extreme.

In this case the fully uncontained design problem can be defined as designing the bridge to satisfy the design specifications and code requirements, with no other constraints. The methodology deployed in this open ended design will have to determine the bridge type, materials, bridge geometry and member properties. The bridge type may be one of the known bridge types or it may be a new, as yet unknown, bridge type and material.

In the fully constrained optimization problem the bridge type, configuration and geometry are known and the methodology is used to determine the member section properties. It is clear that this is an optimization problem, and it is a search in a finite dimensional vector space. This problem has at least one, and possibly multiple optimum solutions.

The fully unconstrained design takes place in a high dimensional vector space. If the search is in an infinite dimensional vector space, then there may be an infinite number of solutions and no optimum solution. As we constrain the problem, we reduce the dimension of the space where the search for the solution takes place. In the limit we have the fully constrained optimization problem.

Genetic algorithm and evolutionary methods have been extensively used at the lower end of this spectrum, in the fully constrained optimization problems. However, genetic algorithm is very versatile and powerful; it can be formulated for use in the unconstrained design problem. Two examples of the application of genetic algorithm in the unconstrained design are given in (Shrestha, 1998; Shrestha and Ghaboussi, 1997; 1998) and (Raich, 1998, Raich and Ghaboussi, 1997a, 1997b, 1998, 2000a, 2000b, 2001).

In (Shrestha 1998; Shrestha and Ghaboussi 1997, 1998) a truss is to be designed for a span to carry a given load. The only constraint is that the truss has to be within a rectangular region of the space with a horizontal dimension of the span and a specified height. No other constraints are applied. A special formulation is developed for this problem that allows the evolution of nodes and members of the truss. Creative designs evolve after about 5000 generations. The evolution goes through three distinct stages. In the first stage of about 1000

generations, search for a truss configuration takes place. In this stage, the cost and the disorder in the system remain high. The second stage begins with the beginning of finding of a possible solution and it is characterized by a rapid decline in the cost and the disorder. In the third stage the configuration is finalized and the members are optimized. The cost and the disorder decline slowly in the third stage. Since the configuration changes little in the third stage, genetic algorithm is in fact solving an optimization problem.

In the above mentioned papers by Raich and Raich and Ghaboussi genetic algorithm is formulated for solving the unconstrained design problem of a multistory plane frame to carry the dead loads, live loads and wind load, and to satisfy all the applicable code requirements. The only constraints in this problem are the specified horizontal floors and the rectangular region bounding the frame. Except the horizontal members supporting the floors, no other restrictions were imposed on the location and orientation of the members. A special form of genetic algorithm that allows for implicit representation and redundancy in the genes was developed and used in this problem (see Raich and Ghaboussi, 1997a). It is important to note that since the problem has multiple solutions, each time the process starts from a random initial condition it may lead to a completely different solution.

This formulation led to evolution of highly creative and unexpected frame designs. In some designs the main load carrying mechanism was through an arch embedded in the frame, while in others it was through a truss that was embedded in the frame. It is known that arches and trusses are more efficient load carrying structures than the frames.

In both of the examples cited above, genetic algorithm was formulated and used as a direct method of solving the inverse problem of design, not as a method of optimization. As was mentioned earlier, design includes optimization in a wider search for solution. Both examples also demonstrate that evolutionary methods are capable of creative designs. One of the objectives of undertaking these research projects was to find out whether evolution, in the form used in genetic algorithm, was capable of creative design. The answer was a resounding affirmative.

An important point in the two examples of engineering design was the role of redundancy in genetic codes. Although simple genetic algorithm was used in the first example, the formulation was such that it resulted in large segments of genetic code being redundant at any generation. In the second example Implicit Redundant Representation Genetic Algorithm (see Raich and Ghaboussi, 1997) was used, and this algorithm allows redundant segments in the genetic code. Redundancy appears to have played an important role in the successful application of genetic algorithm to these difficult inverse problems.

IRRGA has also been applied in form finding and design of tensegrity structures (see Chang, 2006). The stable geometry of a tensegrity structure is the result of internal tensile and compressive member forces balancing each other

and maintaining the geometry of the structure. Some forms of tensegrity structures are well known. Generally, finding the form of tensegrity structures is a difficult task and no methods are currently available for form finding and design of this class of structures. IRRGA was successfully applied to this task and details are available in (Chang, 2006).

References

Bani-Hani, K. and Ghaboussi, J. (1998a). Nonlinear structural control using neural networks. *Journal of Engineering Mechanics Division*, *ASCE* 124: 319-327.

Bani-Hani, K. and Ghaboussi, J. (1998b). Neural networks for structural control of a benchmark problem: Active tendon system. *International Journal for Earthquake Engineering and Structural Dynamics* 27:1225-1245.

Bani-Hani, K., Ghaboussi, J. and Schneider, S.P. (1999a). Experimental study of identification and structural control using neural network: I. Identification. *International Journal for Earthquake Engineering and Structural Dynamics* 28:995-1018.

Bani-Hani, K., Ghaboussi, J. and Schneider, S.P. (1999b). Experimental study of identification and structural control using neural network: II. Control. *International Journal for Earthquake Engineering and Structural Dynamics* 28:1019-1039.

Chang, Y.-K. (2006). *Evolutionary Based Methodology for Form-finding and Design of Tensegrity Structures*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois.

Chou, J.-H. (2000). *Study of Condition Monitoring of Bridges Using Genetic Algorithm.* PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois.

Chou, J.-H. and Ghaboussi, J. (1997). Structural damage detection and identification using genetic algorithm. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Chou, J.-H. and Ghaboussi, J. (1998). Studies in bridge damage detection using genetic algorithm. *Proceedings, Sixth East Asia-Pacific Conference on Structural Engineering and Construction (EASEC6)*. Taiwan.

Chou J. H. and Ghaboussi, J. (2001). Genetic algorithm in structural damage detection. *International Journal of Computers and Structures* 79:1335-1353.

Chou, J.-H., Ghaboussi, J. and Clark, R. (1998). Application of neural networks to the inspection of railroad rail. *Proceedings, Twenty-Fifth Annual Conference on Review of Progress in Quantitative Nondestructive Evaluation*. Snowbird Utah.

Fu, Q., Hashash, Y. M. A., Jung, S., and Ghaboussi, J. (2007). Integration of laboratory testing and constitutive modeling of soils. *Computer and Geotechnics* 34: 330-345.

Ghaboussi, J. (1992a). Potential applications of neuro-biological computational models in geotechnical engineering. *Proceedings, International Conference on Numerical Models in Geotechnical Engineering, NUMOG IV*. Swansea, UK

Ghaboussi, J. (1992b). Neuro-biological computational models with learning capabilities and their applications in geomechanical modeling. *Proceedings, Workshop on Recent Accomplishments and Future Trends in Geomechanics in the 21st Century*. Norman, Oklahoma.

Ghaboussi, J. (2001) Biologically Inspired Soft Computing Methods in Structural Mechanics and Engineering, *International Journal of Structural Engineering and Mechanics* 11:485-502.

Ghaboussi, J. and Banan, M.R. (1994). Neural networks in engineering diagnostics, *Proceedings, Earthmoving Conference, Society of Automotive Engineers*, Peoria, Illinois, SAE Technical Paper No. 941116.

Ghaboussi, J., Banan, M.R. and Florom, R. L. (1994). Application of neural networks in acoustic wayside fault detection in railway engineering. *Proceedings, World Congress on Railway Research*. Paris, France.

Ghaboussi, J., Garrett, Jr., J.H. and Wu, X. (1990). Material modeling with neural networks. *Proceedings, International Conference on Numerical Methods in Engineering*: *Theory and Applications*. Swansea, U.K.

Ghaboussi, J., Garrett, J.H. and Wu, X. (1991). Knowledge-based modeling of material behavior with neural networks. *Journal of Engineering Mechanics Division*, *ASCE* 117:132 – 153.

Ghaboussi, J. and Joghataie, A. (1995). Active control of structures using Neural networks. *Journal of Engineering mechanics Division*, *ASCE* 121: 555- 567.

Ghaboussi, J., Lade, P.V. and Sidarta, D.E. (1994). Neural network based modeling in geomechanics. *Proceedings, International Conference on Numerical Methods and Advances in Geomechanics*.

Ghaboussi, J. and Lin, C.-C.J. (1998). A new method of generating earthquake accelerograms using neural networks. *International Journal for Earthquake Engineering and Structural Dynamics* 27:377-396.

Ghaboussi, J., Pecknold, D.A., Zhang, M. and HajAli, R. (1998). Auto-progressive training of neural network constitutive models. *International Journal for Numerical Methods in Engineering* 42:105-126.

Ghaboussi, J., Pecknold D.A. and Zhang, M. (1998). Autoprogressive training of neural networks in complex systems. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Ghaboussi, J. and Sidarta, D.E. (1998). A new nested adaptive neural network for modeling of constitutive behavior of materials. *International Journal of Computer and Geotechnics* 22: 29-51.

Ghaboussi J. and Wu, X. (1998). Soft computing with neural networks for engineering applications: Fundamental issues and adaptive approaches. *International Journal of Structural Engineering and Mechanics* 8: 955 -969.

Ghaboussi, J., Zhang, M., Wu X. and Pecknold, D.A. (1997). Nested adaptive neural networks: A new architecture. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Gashash, Y.M.A., Jung, S. and Ghaboussi, J. (2004). Numerical implementation of a neural networks based material model in finite element. *International Journal for Numerical Methods in Engineering* 59:989-1005.

Hashash, Y.M.A., Marulanda, C. Ghaboussi, J. and Jung, S. (2003). Systematic update of a deep excavation model using field performance data", *Computers and Geotechnics* 30:477-488.

Hashash, Y.M.A., Marulando, C., Ghaboussi, J. and Jung, S. (2006). Novel approach to integration of numerical modeling and field observation for deep excavations. *Journal of Geotechnical and Geo-environmental Engineering*, *ASCE* 132: 1019-1031.

Hashash, Y.M.A., Fu, Q.-W., Ghaboussi, J., Lade, P. V. and Saucier, C. (2009). Inverse analysis based interpretation of sand behavior from triaxial shear tests subjected to full end restraint. *Canadian Geotechnical Journal*, in press.

Joghataie, A., Ghaboussi, J. and Wu, X. (1995). Learning and architecture determination through automatic node generation. *Proceedings, International Conference on Artificial neural Networks in Engineering. ANNIE*. St Louis

Jung. S.-M. and Ghaboussi, J. (2006a). Neural network constitutive model for rate-dependent materials, *Computer and Sstructures* 84:955-963.

Jung. S-M and Ghaboussi, J. (2006b). Characterizing rate-dependent material behaviors in self-learning simulation. *Computer Methods in Applied Mechanics and Engineering* 196:608-619.

Jung, S.-M., Ghaboussi, J. and Kwon, S.-D. (2004). Estimation of aero-elastic parameters of bridge decks using neural networks. *Journal of Engineering Mechanics Division, ASCE* 130:1356 – 1364.

Jung S-M, Ghaboussi J. and Marulanda, C. (2007). Field calibration of time-dependent behavior in segmental bridges using self-learning simulations. *Engineering Structures* 29:2692-2700.

Kaklauskas, G. and Ghaboussi, J. (2000). Concrete stress-strain relations from R/C beam tests. *Journal of Structural Engineering*, *ASCE* 127, No. 1.

Kim, Y.-J. (2000). *Active Control of Structures Using Genetic Algorithm*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois

Kim, Y. J. and Ghaboussi, J. (1999). A new method of reduced order feedback control using genetic algorithm. *International Journal for Earthquake Engineering and Structural Dynamics* 28: 193-212.

Kwon T.-H (2006) *Minimally Invasive Characterization and Intraocular Pressure Measurement via Numerical Simulation of Human Cornea*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois.

Kwon, T.-H. Ghaboussi, J., Pecknold, D. A. and Hashash, Y.M.A. (2008). Effect of cornea stiffness on measured intraocular pressure. *Journal of Biomechanics* 41:1707-1713.

Kwon, T.-H. Ghaboussi, J., Pecknold, D. A. and Hashash, Y.M.A. (2009). Role of cornea biomechanical properties in applanation tonometry measurements. *Journal of Refractive Surgery*, in press

Lin, C.-C. J. (1999). *A neural network based methodology for generating spectrum compatible earthquake accelerograms*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois.

Lin C.-C. J. and Ghaboussi, J. (2001). Generating multiple spectrum compatible accelrograms using neural networks. *International Journal Earthquake Engineering and Structural Dynamics* 30:1021-1042.

Limsamphancharon, N. (2003) *Condition Monitoring of Structures by Using Ambient Dynamic Responses*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign.Urbana, Illinois.

Nikzad, K., Ghaboussi, J. and Paul, S.L. (1996). A study of actuator dynamics and delay compensation using a neuro-controller. *Journal of Engineering Mechanics Division*, *ASCE* 122:966-975.

Raich, Anne M. (1998). *An Evolutionary Based Methodology for Representing and Evolving Structural Design Solutions*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois.

Raich, A. M. and Ghaboussi, J. (1997a). Implicit Representation in Genetic Algorithm Using Redundancy. *International Journal of Evolutionary Computing* 5, No. 3.

Raich, A. M. and Ghaboussi, J. (1997b). Autogenesis and Redundancy in Genetic Algorithm Representation. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Raich, A.M. and Ghaboussi, J. (1998). Evolving design solutions by transforming the design environment. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Raich, A. M. and Ghaboussi, J. (2000a). Evolving structural design solutions using an implicit redundant genetic algorithm. *Journal of Structural and Multi-disciplinary Optimization* 20:222-231.

Raich, A. M. and Ghaboussi, J., (2000b). Applying the implicit redundant representation genetic algorithm. In Lance Chambers , ed, *Practical Handbook of Genetic Algorithm, vol. 1. Applications,* Ch.9 Unstructured Problem Domain. Chapman&Hall/CRC Publishers.

Raich, A.M., and Ghaboussi, J. (2001) "Evolving the topology and geometry of frame structures during optimization. *Structural Optimization* 20:222-231.

Sidarta, D.E. and Ghaboussi, J. (1998). Modeling constitutive behavior of materials from non-uniform material tests. *International Journal of Computer and Geotechnics* 22, No. 1.

Shrestha, S.M. (1998). *Genetic Algorithm Based Methodology for Unstructured Design of Skeletal Structures*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois

Shrestha, S.M. and Ghaboussi, J. (1997). Genetic algorithm in structural shape design and optimization. *Proceedings, 7th International Conference on Computing in Civil and Building Engineering (ICCCBE-VII)*. Seoul, South Korea

Shrestha, S.M. and Ghaboussi, J. (1998). Evolution of optimal structural shapes using genetic algorithm. *Journal of Structural Engineering*, *ASCE*. 124:1331 -1338.

Wu, X. (1991). *Neural Network Based Material Modeling*. PhD thesis, Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign. Urbana, Illinois

Wu, X. and Ghaboussi, J. (1993). Modeling unloading and cyclic behavior of concrete with adaptive neural networksn. *Proceedings, APCOM'93, 2nd Asian-Pacific Conference on Computational Mechanics*.Sydney Australia

Wu, X., Ghaboussi, J. and Garrett, J.H. (1992). Use of neural networks in detection of structural damage. *Computers and Structures*, 42:649-660.

Yun, G.-J., (2006) *Modeling of Hysteretic Behavior of Beam-column Connections Based on Self-learning Simulations*. PhD thesis, Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Yun G, J, Ghaboussi J. and Elnashai, A.S. (2008a). A New Neural Network Based Model for Hysteretic Behavior of Materials. *International Journal for Numerical Methods in Engineering* 73:447-469.

Yun G.J, Ghaboussi J. and Elnashai, A.S. (2008b). Self-learning simulation method for inverse nonlinear modeling of cyclic behavior of connections, *Computer Methods in Applied Mechanics and Engineering* 197:2836-2857.

Yun G.J. Ghaboussi J. and Elnashai, A.S. (2008c). A design-based hysteretic model for beam-column connections. *International Journal of Earthquake Engineering and Structural Dynamics*. 37:535-555.

Zhang, M. (1996). *Determination of neural network material models from structural tests.* PhD thesis, Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign. Urbana, Illinois.

and maintaining the geometry of the structure. Some forms of tensegrity structures are well known. Generally, finding the form of tensegrity structures is a difficult task and no methods are currently available for form finding and design of this class of structures. IRRGA was successfully applied to this task and details are available in (Chang, 2006).

References

Bani-Hani, K. and Ghaboussi, J. (1998a). Nonlinear structural control using neural networks. *Journal of Engineering Mechanics Division*, *ASCE* 124: 319-327.

Bani-Hani, K. and Ghaboussi, J. (1998b). Neural networks for structural control of a benchmark problem: Active tendon system. *International Journal for Earthquake Engineering and Structural Dynamics* 27:1225-1245.

Bani-Hani, K., Ghaboussi, J. and Schneider, S.P. (1999a). Experimental study of identification and structural control using neural network: I. Identification. *International Journal for Earthquake Engineering and Structural Dynamics* 28:995-1018.

Bani-Hani, K., Ghaboussi, J. and Schneider, S.P. (1999b). Experimental study of identification and structural control using neural network: II. Control. *International Journal for Earthquake Engineering and Structural Dynamics* 28:1019-1039.

Chang, Y.-K. (2006). *Evolutionary Based Methodology for Form-finding and Design of Tensegrity Structures*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois.

Chou, J.-H. (2000). *Study of Condition Monitoring of Bridges Using Genetic Algorithm.* PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois.

Chou, J.-H. and Ghaboussi, J. (1997). Structural damage detection and identification using genetic algorithm. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Chou, J.-H. and Ghaboussi, J. (1998). Studies in bridge damage detection using genetic algorithm. *Proceedings, Sixth East Asia-Pacific Conference on Structural Engineering and Construction (EASEC6)*. Taiwan.

Chou J. H. and Ghaboussi, J. (2001). Genetic algorithm in structural damage detection. *International Journal of Computers and Structures* 79:1335-1353.

Chou, J.-H., Ghaboussi, J. and Clark, R. (1998). Application of neural networks to the inspection of railroad rail. *Proceedings, Twenty-Fifth Annual Conference on Review of Progress in Quantitative Nondestructive Evaluation*. Snowbird Utah.

Fu, Q., Hashash, Y. M. A., Jung, S., and Ghaboussi, J. (2007). Integration of laboratory testing and constitutive modeling of soils. *Computer and Geotechnics* 34: 330-345.

Ghaboussi, J. (1992a). Potential applications of neuro-biological computational models in geotechnical engineering. *Proceedings, International Conference on Numerical Models in Geotechnical Engineering, NUMOG IV*. Swansea, UK

Ghaboussi, J. (1992b). Neuro-biological computational models with learning capabilities and their applications in geomechanical modeling. *Proceedings, Workshop on Recent Accomplishments and Future Trends in Geomechanics in the 21st Century*. Norman, Oklahoma.

Ghaboussi, J. (2001) Biologically Inspired Soft Computing Methods in Structural Mechanics and Engineering, *International Journal of Structural Engineering and Mechanics* 11:485-502.

Ghaboussi, J. and Banan, M.R. (1994). Neural networks in engineering diagnostics, *Proceedings, Earthmoving Conference, Society of Automotive Engineers*, Peoria, Illinois, SAE Technical Paper No. 941116.

Ghaboussi, J., Banan, M.R. and Florom, R. L. (1994). Application of neural networks in acoustic wayside fault detection in railway engineering. *Proceedings, World Congress on Railway Research*. Paris, France.

Ghaboussi, J., Garrett, Jr., J.H. and Wu, X. (1990). Material modeling with neural networks. *Proceedings, International Conference on Numerical Methods in Engineering*: *Theory and Applications*. Swansea, U.K.

Ghaboussi, J., Garrett, J.H. and Wu, X. (1991). Knowledge-based modeling of material behavior with neural networks. *Journal of Engineering Mechanics Division*, *ASCE* 117:132 – 153.

Ghaboussi, J. and Joghataie, A. (1995). Active control of structures using Neural networks. *Journal of Engineering mechanics Division*, *ASCE* 121: 555- 567.

Ghaboussi, J., Lade, P.V. and Sidarta, D.E. (1994). Neural network based modeling in geomechanics. *Proceedings, International Conference on Numerical Methods and Advances in Geomechanics*.

Ghaboussi, J. and Lin, C.-C.J. (1998). A new method of generating earthquake accelerograms using neural networks. *International Journal for Earthquake Engineering and Structural Dynamics* 27:377-396.

Ghaboussi, J., Pecknold, D.A., Zhang, M. and HajAli, R. (1998). Auto-progressive training of neural network constitutive models. *International Journal for Numerical Methods in Engineering* 42:105-126.

Ghaboussi, J., Pecknold D.A. and Zhang, M. (1998). Autoprogressive training of neural networks in complex systems. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Ghaboussi, J. and Sidarta, D.E. (1998). A new nested adaptive neural network for modeling of constitutive behavior of materials. *International Journal of Computer and Geotechnics* 22: 29-51.

Ghaboussi J. and Wu, X. (1998). Soft computing with neural networks for engineering applications: Fundamental issues and adaptive approaches. *International Journal of Structural Engineering and Mechanics* 8: 955 -969.

Ghaboussi, J., Zhang, M., Wu X. and Pecknold, D.A. (1997). Nested adaptive neural networks: A new architecture. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Gashash, Y.M.A., Jung, S. and Ghaboussi, J. (2004). Numerical implementation of a neural networks based material model in finite element. *International Journal for Numerical Methods in Engineering* 59:989-1005.

Hashash, Y.M.A., Marulanda, C. Ghaboussi, J. and Jung, S. (2003). Systematic update of a deep excavation model using field performance data", *Computers and Geotechnics* 30:477-488.

Hashash, Y.M.A., Marulando, C., Ghaboussi, J. and Jung, S. (2006). Novel approach to integration of numerical modeling and field observation for deep excavations. *Journal of Geotechnical and Geo-environmental Engineering*, *ASCE* 132: 1019-1031.

Hashash, Y.M.A., Fu, Q.-W., Ghaboussi, J., Lade, P. V. and Saucier, C. (2009). Inverse analysis based interpretation of sand behavior from triaxial shear tests subjected to full end restraint. *Canadian Geotechnical Journal*, in press.

Joghataie, A., Ghaboussi, J. and Wu, X. (1995). Learning and architecture determination through automatic node generation. *Proceedings, International Conference on Artificial neural Networks in Engineering. ANNIE*. St Louis

Jung. S.-M. and Ghaboussi, J. (2006a). Neural network constitutive model for rate-dependent materials, *Computer and Sstructures* 84:955-963.

Jung. S-M and Ghaboussi, J. (2006b). Characterizing rate-dependent material behaviors in self-learning simulation. *Computer Methods in Applied Mechanics and Engineering* 196:608-619.

Jung, S.-M., Ghaboussi, J. and Kwon, S.-D. (2004). Estimation of aero-elastic parameters of bridge decks using neural networks. *Journal of Engineering Mechanics Division, ASCE* 130:1356 – 1364.

Jung S-M, Ghaboussi J. and Marulanda, C. (2007). Field calibration of time-dependent behavior in segmental bridges using self-learning simulations. *Engineering Structures* 29:2692-2700.

Kaklauskas, G. and Ghaboussi, J. (2000). Concrete stress-strain relations from R/C beam tests. *Journal of Structural Engineering*, *ASCE* 127, No. 1.

Kim, Y.-J. (2000). *Active Control of Structures Using Genetic Algorithm*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois

Kim, Y. J. and Ghaboussi, J. (1999). A new method of reduced order feedback control using genetic algorithm. *International Journal for Earthquake Engineering and Structural Dynamics* 28: 193-212.

Kwon T.-H (2006) *Minimally Invasive Characterization and Intraocular Pressure Measurement via Numerical Simulation of Human Cornea*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois.

Kwon, T.-H. Ghaboussi, J., Pecknold, D. A. and Hashash, Y.M.A. (2008). Effect of cornea stiffness on measured intraocular pressure. *Journal of Biomechanics* 41:1707-1713.

Kwon, T.-H. Ghaboussi, J., Pecknold, D. A. and Hashash, Y.M.A. (2009). Role of cornea biomechanical properties in applanation tonometry measurements. *Journal of Refractive Surgery*, in press

Lin, C.-C. J. (1999). *A neural network based methodology for generating spectrum compatible earthquake accelerograms*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois.

Lin C.-C. J. and Ghaboussi, J. (2001). Generating multiple spectrum compatible accelrograms using neural networks. *International Journal Earthquake Engineering and Structural Dynamics* 30:1021-1042.

Limsamphancharon, N. (2003) *Condition Monitoring of Structures by Using Ambient Dynamic Responses*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign.Urbana, Illinois.

Nikzad, K., Ghaboussi, J. and Paul, S.L. (1996). A study of actuator dynamics and delay compensation using a neuro-controller. *Journal of Engineering Mechanics Division*, *ASCE* 122:966-975.

Raich, Anne M. (1998). *An Evolutionary Based Methodology for Representing and Evolving Structural Design Solutions*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois.

Raich, A. M. and Ghaboussi, J. (1997a). Implicit Representation in Genetic Algorithm Using Redundancy. *International Journal of Evolutionary Computing* 5, No. 3.

Raich, A. M. and Ghaboussi, J. (1997b). Autogenesis and Redundancy in Genetic Algorithm Representation. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Raich, A.M. and Ghaboussi, J. (1998). Evolving design solutions by transforming the design environment. *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE*. St. Louis, Missouri.

Raich, A. M. and Ghaboussi, J. (2000a). Evolving structural design solutions using an implicit redundant genetic algorithm. *Journal of Structural and Multi-disciplinary Optimization* 20:222-231.

Raich, A. M. and Ghaboussi, J., (2000b). Applying the implicit redundant representation genetic algorithm. In Lance Chambers , ed, *Practical Handbook of Genetic Algorithm, vol. 1. Applications,* Ch.9 Unstructured Problem Domain. Chapman&Hall/CRC Publishers.

Raich, A.M., and Ghaboussi, J. (2001) "Evolving the topology and geometry of frame structures during optimization. *Structural Optimization* 20:222-231.

Sidarta, D.E. and Ghaboussi, J. (1998). Modeling constitutive behavior of materials from non-uniform material tests. *International Journal of Computer and Geotechnics* 22, No. 1.

Shrestha, S.M. (1998). *Genetic Algorithm Based Methodology for Unstructured Design of Skeletal Structures*. PhD thesis, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign. Urbana, Illinois

Shrestha, S.M. and Ghaboussi, J. (1997). Genetic algorithm in structural shape design and optimization. *Proceedings, 7th International Conference on Computing in Civil and Building Engineering (ICCCBE-VII)*. Seoul, South Korea

Shrestha, S.M. and Ghaboussi, J. (1998). Evolution of optimal structural shapes using genetic algorithm. *Journal of Structural Engineering*, *ASCE*. 124:1331 -1338.

Wu, X. (1991). *Neural Network Based Material Modeling*. PhD thesis, Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign. Urbana, Illinois

Wu, X. and Ghaboussi, J. (1993). Modeling unloading and cyclic behavior of concrete with adaptive neural networksn. *Proceedings, APCOM'93, 2nd Asian-Pacific Conference on Computational Mechanics*.Sydney Australia

Wu, X., Ghaboussi, J. and Garrett, J.H. (1992). Use of neural networks in detection of structural damage. *Computers and Structures*, 42:649-660.

Yun, G.-J., (2006) *Modeling of Hysteretic Behavior of Beam-column Connections Based on Self-learning Simulations*. PhD thesis, Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Yun G, J, Ghaboussi J. and Elnashai, A.S. (2008a). A New Neural Network Based Model for Hysteretic Behavior of Materials. *International Journal for Numerical Methods in Engineering* 73:447-469.

Yun G.J, Ghaboussi J. and Elnashai, A.S. (2008b). Self-learning simulation method for inverse nonlinear modeling of cyclic behavior of connections, *Computer Methods in Applied Mechanics and Engineering* 197:2836-2857.

Yun G.J. Ghaboussi J. and Elnashai, A.S. (2008c). A design-based hysteretic model for beam-column connections. *International Journal of Earthquake Engineering and Structural Dynamics*. 37:535-555.

Zhang, M. (1996). *Determination of neural network material models from structural tests.* PhD thesis, Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign. Urbana, Illinois.

CHAPTER 5

Selected Problems of Artificial Neural Networks Development

Zenon Waszczyszyn [1,2] and Marek Słoński [2*]

[1] Chair of Structural Mechanics,
Rzeszów University of Technology, Rzeszów, Poland,

[2] Institute for Computational Civil Engineering,
Cracow University of Technology, Kraków, Poland

Abstract. The chapter discusses selected problems of applications of Standard (deterministic) Neural Networks (SNN) but the main attention is focused on Bayesian Neural Networks (BNNs). In Sections 2 and 3 the problems of regression analysis, over-fitting and regularization are discussed basing on two types of network, i.e. Feed-forward Layered Neural Network (FLNN) and Radial Basis Function NN (RBFN). Application of Principal Component Analysis (PCA) is discussed as a method for reduction of input space dimensionality. In Section 4 the application of Kalman filtering to learning of SNNs is presented. Section 5 is devoted to discussion of some basics related to Bayesian inference. Then Maximum Likelihood (ML) and Maximum APosterior (MAP) methods are presented as a basis for formulation of networks SNN-ML and SNN-MAP. A more general Bayesian framework corresponding to formulation of simple, semi-probabilistic network S-BNN, true probabilistic T-BNN and Gaussian Process GP-BNN is discussed. Section 6 is devoted to the analysis of four study cases, related mostly to the analysis of structural engineering and material mechanics problems.

1 Introduction

Artificial Neural Networks belong to a group of □*biologically*□*inspired methods* together with fuzzy systems and genetic algorithms (or, more generally, evolutionary algorithms, systems and strategies). These methods are sometimes called *soft*

*Authors would like to acknowledge support from the Polish Ministry of Science and Higher Education Grant "Application of Bayesian neural networks in experimental mechanics of structures and materials", No. N506 1814 33.

computing methods because these methods can also have such features as adaptivity and flexibility to find optimal solutions. That is why soft methods can also be called methods of *computational artificial intelligence*, see Jang et al. (1997). ANNs have turned out to be an especially efficient tool in the analysis of classification and regression problems, i.e. problems which are commonly analyzed in various engineering applications.

ANNs were presented at two CISM Courses devoted either to the analysis and design of structures, see Waszczyszyn (1999), or to the identification problems of structural engineering, see Waszczyszyn and Ziemiański (2005). The corresponding papers on regression problems can be briefly characterized as the mapping of input data onto predicted output variables.

The presented Chapter is called "Selected Problems of ANN Development" and it is, in fact, a continuation of two previous cycles of lectures presented in Udine at CISM Advanced Courses in 1998 and 2002. From the viewpoint of selected topics, the Chapter deals first with some problems of standard, deterministic neural networks. Attention is focused on Feed-forward-Layered Neural Network (FLNN) and Radial Basis Function NN (RBFN). These networks are suitable for the analysis of regression problems which are discussed in this Chapter.

FLNN and *RBFN* are sometimes called *standard networks* since they give a numerical solution in form of real numbers, and computations are carried out on the base of the *error minimizing paradigm*, see Tipping (2004). Fulfilling this paradigm needs the application of various learning methods which explore different iterative techniques. In a special case of linear in weights RBFN the learning method can be reduced to the solution of a linear set of algebraic equations. Because of the minimization paradigm also the networks trained by means of stochastic Kalman filtering are treated as standard, deterministic networks.

The second part of the Chapter is devoted to probabilistic networks supported on the Bayesian *inference paradigm*. This approach explores the Bayes' theorem, which in general does not require iterative techniques, and is based either on the integration over weights (*marginalization principle*) or on the *random process* interpolation method.

The Bayesian approach was introduced to neural networks quite early due to Buntine and Weigend (1991) but mainly the paper by MacKay (1992) and report written by Neal (1992) inspired new research and applications. After the first book on BNNs was written by Bishop (1995) many new books and state-of-the-art papers were published. From among them we quote here only selected books and reviews which seem to be accessible for students and engineers and useful for extending their interest in BNNs. In recent years there were published books by MacKay (2003), 6-th printing in 2007, Bishop (2006), Rasmussen and Williams (2006). From among many reviews those by Lampinen and Vehtari (2001) and Tipping (2004) are especially worth recommending for reading.

In the frame of the Bayesian framework the Bayesian Neural Network (BNN) was formulated in which the standard, deterministic network is used as a computational interface. In this Chapter, we apply only FLNN and RBFN networks to carry out computation needed to explore the Bayesian approach and improve the analysis of SNN networks. BNN are now in the centre of research interest so their foundations are briefly discussed in the Chapter.

Special attention is paid to discussions of computational methods needed in the Bayesian approach. This concerns a deterministic, in fact, method of *Maximum Likelihood*, which fully corresponds to the minimization method of Least-Square error. A much more general method of the *Maximum a Posterior* is extensively discussed as well. What seems especially interesting is the application of the *Maximum Marginalized Posterior* as a new criterion for neural network design.

In the Chapter we focus on selected problems concerning standard neural networks. We discuss in short the basic problem of the over-fitting phenomenon and a possibility of controlling it by the weight-decay regularization method. Then we focus on design of neural networks. The most important question of selection of a neural network optimal model is discussed on the base of the cross-validation method and from the viewpoint of a possibility of improvement of this technique due to application of the Bayesian approach. We focus also on the possibility of reduction of the input space dimensionality by the application of the *Principal Component Analysis*.

From among extended literature devoted to neural networks we selected the book by Haykin (1999) as a base for preparing the first part of the Chapter. The second part devoted to the Bayesian neural network is based on the book by Bishop (2006).

The content of subsequent Sections is mainly illustrated by the analysis of a simple study case related to synthetic sinusoidal data. Without loss of generality we limited our considerations to the analysis of one-dimensional regression problem (single output) and D-dimensional input space. In the frame of such assumptions the problems discussed in Sections 2-5 are illustrated in Section 6 on examples of engineering applications, described in papers by the authors of the Chapter and their associates.

2 Regression, Over-Fitting and Regularization

2.1 Regression function and regressive models

Regression corresponds to mapping of input variables x_j onto output variables which should be equal targets t_i. For the sake of clarity we have assumed one dimensional output for considering the definition of the target field of the output data:

$$t(\mathbf{x}) = h(\mathbf{x}) + \varepsilon(\mathbf{x}), \tag{1}$$

where: $h(\mathbf{x})$ – scalar deterministic function, $\varepsilon(\mathbf{x})$ – field of random noise, see Figure 1a.

In what follows we focus on data completed of data points, corresponding to events, observations or measurements, composed as pairs of input/output:

$$\mathscr{D} = \{(\mathbf{x},t)^n\}_{n=1}^N \equiv \{\mathbf{x}^n,t^n\}_{n=1}^N. \tag{2}$$

The empirical knowledge, represented by set $\mathscr{D}$, is encoded in a numerical model by the vector of parameters $\mathbf{w}$, i.e. $\mathscr{D} \to \mathbf{w}$ so the predicted output (numerical approximation) is $y(\mathbf{x};\mathbf{w})$. On the base of these remarks two regressive models are formulated, see Figure 1 taken from Haykin (1999), p.85.

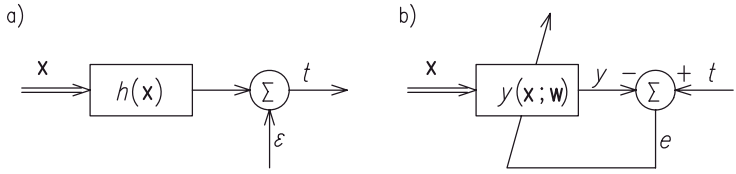


Figure 1. Regressive models: a) Mathematical model, b) Physical (neural network) model

Simple numerical models can be considered as deterministic algorithms or systems such as Artificial Neural Networks (ANNs) in which a computed solution $y(\mathbf{x}^n;\mathbf{w})$ is close to the target output t^n, see Figure 1b:

$$t^n = y(\mathbf{x};\mathbf{w}) + e^n, \tag{3}$$

i.e. the computed outputs are perturbed by computational and model errors e^n. In ANNs weight vector $\mathbf{w}$ is composed of synaptic weights and biases

$$\mathbf{w} = \{\mathbf{w}\}_{i=1}^W, \tag{4}$$

where: W – dimension of the weight space.

The vector of weights is computed by means of the training (learning) process applying a supervised learning method. Both the training processes and corresponding learning methods strongly depend on the regression model applied. From among many models we focus on the linear models which are based either on the polynomials or the Gaussian Radial Basis Functions.

2.2 Polynomial approximation of the regression function

Values of parameters w_k can be computed applying the least square error function

$$E_{\mathscr{D}}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t^n - y(\mathbf{x}^n;\mathbf{w})\}^2. \tag{5}$$

Optimal values of weights w_k^{LS} are computed by means of the Least Square method (marked by super- or subscripts LS or LSM) which is related to searching the minimal value of the function (5):

$$\min_{\mathbf{w}} E_{\mathscr{D}}(\mathbf{w}) \equiv \frac{\partial E_{\mathscr{D}}(\mathbf{w})}{\partial w_k} = 0 \text{ for } k = 0,\ldots,W-1 \rightarrow \mathbf{w}_{\mathrm{LS}}. \tag{6}$$

For the sake of clarity let us assume a one-dimensional input space $D = 1$ and polynomial regression function of the K-th order

$$y(x;\mathbf{w}) = w_0 + w_1 x + w_2 (x)^2 + \cdots + w_K (x)^K \equiv \sum_{k=0}^{K=W-1} w_k (x)^k. \tag{7}$$

Substitution of (7) to criterion (6) leads to the linear set of $W = K+1$ equations from which weights w_i can be computed for $i = 1,\ldots,W$.

Let us discuss a simple numerical example of sinusoidal curve fitting, discussed in many books and papers, cf. e.g. books by Bishop (1995, 2006).

Example 2.1 taken from Bishop (2006), p. 8. The mathematical regression function is the sin curve $h(x) = \sin 2\pi x$ for $x \in [0.0, 1.0]$, marked in Figure 2 by the broken line. Let us consider the case we observe $N = 10$ noisy patterns, see Figure 2a.

It is assumed that the noisy patterns are generated by applying the curve $h(x)$ as the mean value perturbed by the Gaussian noise $\varepsilon(x)$ with the probability distribution of the mean zero and variance σ^2, i.e. $p(\varepsilon^n) = \mathscr{N}(0|\sigma^2)$, see (94). Assuming polynomials (7) of the order $K = 1,3,9$ the corresponding $W = 2,4$ and 10 linear equations of LS method were solved. The values of weights $w_k^{\mathrm{LS}}(K;N,\ln\lambda = -\infty)$ are listed in Table 1.

The sequence of columns 2 – 5 corresponds to the transition from simple to complex polynomial models. Very simple models $K = 0,1$ are related to linear regression which gives a declined line shown in Figure 2b. The third order polynomial leads to quite good fitting to the mathematical regression curve $h(x) = \sin 2\pi x$.

The polynomial of order $K = 9$ matches exactly all the $N = 10$ patterns but the over-fitting phenomenon occurs. This means that the polynomial curve has great oscillations and large values of curvatures.

The over-fitting phenomenon is a well-known problem of regression models which should be formulated in-between the simple and complex models. In case of simple models the accuracy of the approximation can be poor but the polynomial curves are smooth (regular), cf. cases $K = 0,1,2,3$. The increase of the number of model parameters w_k leads to models whose fitting the patterns is better but the over-fitting can occur. What seems an acceptable model is one with $K = 3$, which corresponds to the polynomial of the third order, as shown in Figure 2c.

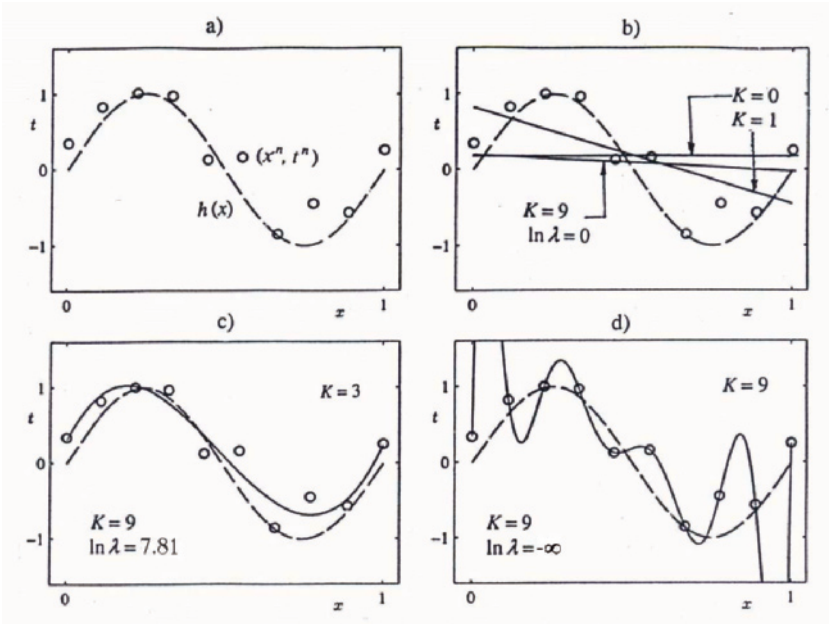


Figure 2. a) Mathematical regression curve $h(x) = \sin 2\pi x$ used for generating $N = 10$ patterns; b, c, d) Fitting curves for polynomials of orders $K = 0, 1, 3$ and 9

Table 1. Weights $w_k^\star(K; N, \ln\lambda)$, where $\star$ = LS or PLS, for the polynomials of order K = 0, 1, 3, 9, for the number of patterns $N = 10$ and different values of the regularization parameter λ

w_k^*	K= 0	K= 1	K = 3	K = 9		
	$\ln\lambda = -\infty$			$\ln\lambda = -\infty$	$\ln\lambda = -7.81$	$\ln\lambda = 0$
1	2	3	4	5	6	7
w_0^*	0.19	0.82	0.3	0.3	0.35	0.13
w_1^*		1.27	8.0	232.4	4.74	0.05
w_2^*			−25.4	−5321.8	−0.77	−0.06
w_3^*			17.3	48568.3	−31.97	−0.05
w_4^*				−231639.3	−3.89	−0.03
w_5^*				640042.3	55.28	−0.02
w_6^*				−1061899.5	41.32	−0.01
w_7^*				1042400.2	−45.95	−0.00
w_8^*				−5576823.0	−91.53	0.00
w_9^*				125201.4	72.68	0.01

The increase of parameter values causes an increase of the model complexity, cf. Table 1. This affects the sensitivity of the model to prediction of new patterns, which are placed in-between the training patterns.

There are many ways to control the over-fitting. One of them corresponds to keeping the ratio of the number of patterns and model parameters $N/W >> 1$. The suggested values of the optimal ratio of N/W is about 10, see e.g. Haykin (1999), p. 207-209. The commonly used technique for controlling the over-fitting is the weight-decay regularization technique.

Coming back to the illustrative example discussed above, the polynomial of the ninth order is used for $N = 100$ patterns. The Ratio $N/W \approx 10$ permits overcoming the over-fitting and improves significantly the accuracy of approximation, see Figure 3.

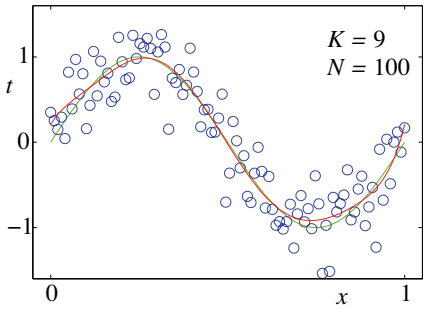


Figure 3. Reduction of over-fitting by increase of data set to N = 100 and applying approximation by polynomial of the order K = 9

2.3 Controlling the over-fitting by weight-decay regularization

Regularization can be involved by adding a penalty term to the error function $E_{\mathscr{D}}(\mathbf{w})$ in order to discourage the weight parameters from reaching large values. The penalty term corresponds to the weight-decay regularization function $E_{\mathrm{W}}(\mathbf{w})$. The modified cost function takes the form of penalized error function $E_{\mathrm{F}}(\mathbf{w})$

$$E_{\mathrm{F}}(\mathbf{w}) = E_{\mathscr{D}}(\mathbf{w}) + E_{\mathrm{W}}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t^n - y(\mathbf{x}^n;\mathbf{w})\}^2 + \frac{\lambda}{2}||\mathbf{w}||^2, \tag{8}$$

where: $E_{\mathrm{F}}(\mathbf{w})$ – full form of the error cost function, $E_{\mathrm{W}}(\mathbf{w})$ – weight-decay regularization function, λ – regularization parameter, $||\mathbf{w}||^2 = \mathbf{w}^{\mathrm{T}}\mathbf{w} = w_0^2 + w_1^2 + \ldots + w_K^2$ – weight measure.

The optimality criterion (6) is used in the form:

$$\min_{\mathbf{w}} E_{\mathrm{F}}(\mathbf{w}) \equiv \frac{\partial E_F(\mathbf{w})}{\partial w_k} + \lambda w_k = 0 \text{ for } k = 0,\ldots,W-1 \rightarrow \mathbf{w}_{\mathrm{PLS}}. \tag{9}$$

The term λw_k stabilizes the solution of the linear set of algebraic equations, where $\mathbf{w}_{\mathrm{PLS}}(\{\mathbf{x}\};\lambda)$ corresponds to an optimal solution obtained by means of the *Penalized Least Square* method (PLS).

The application of regularization parameter $\lambda \geq 0$ enables cancelling the overfitting also for the case $W = N$. The results of computation of $\mathbf{w}_{\mathrm{PLS}}$ are shown for the ninth order of approximation polynomial, see columns 5–7 in Table 1. The case $\ln\lambda = -\infty$ corresponds to $\lambda = 0$, which means vanishing of the regularization term in the penalized error function (8). The values $\ln\lambda = 7.81$ and $\ln\lambda = 0$ give approximations obtained by the polynomial of orders K = 3, 0 but without application of regularization, see Figures 2c,b, respectively.

Summing up this Point, what is worth emphasizing is the role of the weight-decay regularization which permits the control of solution also for complex numerical models without changing the training set.

2.4 ANNs for regression analysis

As mentioned in Section 1, feed-forward Artificial Neural Networks (ANNs) match especially well the analysis of regression problems. In what follows we quote only two types of ANNs with acronyms FLNN (Feed-forward Layered NN) and RBFN (Radial Basis Function NN). These networks are of deterministic character and they are sometimes called Standard Neural Networks (SNNs). Below, FLNN and RBFN are briefly discussed for the case of the one hidden layer, vector input $\mathbf{x}$ and scalar output y.

Feed-forward Layer Neural Network (FLNN). This type of ANN is known in literature under different acronyms, see e.g. MLP (Multi-Layer Perceptron) in the book by Haykin (1999), BPNN (Back-Propagation NN) in the lecture notes by Waszczyszyn (1999) and FLNN (Feed-Forward Layered NN) in the other book by Haykin (2001). In this Chapter acronym FLNN is used.

The simplest FLNN with H sigmoid neurons in a hidden layer and linear output is shown in Figure 4a. The regression function takes the form

$$y(\mathbf{x};\mathbf{w}) = \sum_{h=0}^{H} w_h^2 F_h\Big(\sum_{j=0}^{D} w_{hj}^1 x_j\Big), \tag{10}$$

where: $\mathbf{x} = \{x_j\}_{j=0}^{D} \in \mathscr{R}^{D+1}$ – input vector, $\mathbf{w} = \{w_i\}_{i=1}^{W} \in \mathscr{R}^{W}$ – weight vector, F_h – activation functions in the hidden layer.

The activation functions used in the network layers are shown in Figure 4a:

a) linear identity function for the output:

$$F_{\mathrm{out}} = v, \tag{11}$$

b) bipolar sigmoid for the hidden layer

$$F_h = \frac{1 - e^{-av}}{1 + e^{-av}}, \text{ where } a > 0. \tag{12}$$

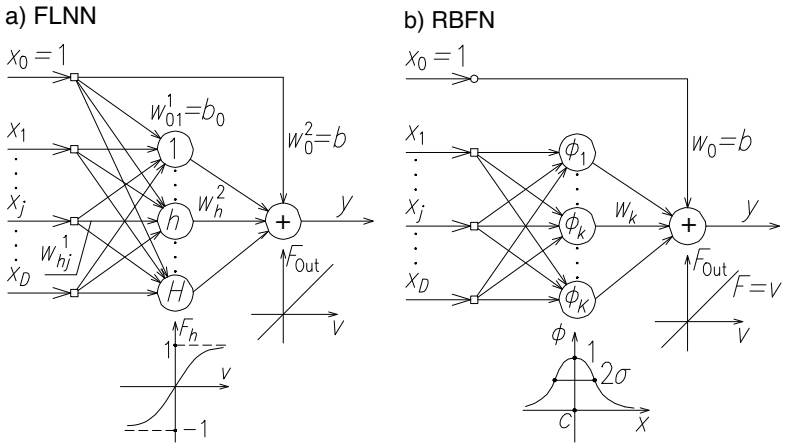


Figure 4. Basic Standard NNs: a) Feed-forward Layer Neural Network (FLNN), b) Radial Basis Function Neural Network (RBFN)

The parameter $a = 1$ is often taken in computations, cf. NN Toolbox by Demuth and Beale (1998) but $a = 2$ is also applied in case of the tanh activation function. The potential corresponding to the layers is

$$1)\ v^1_h = \sum_{j=0}^{D} w^1_{hj} x_j \equiv b_0 + \sum_{j=1}^{D} w^1_{hj} x_j, \tag{13}$$

$$2)\ v_{\text{out}} = b + \sum_{h=1}^{H} F_h(v^1_h) w^2_h. \tag{14}$$

FLNN can be trained (learnt) by different learning methods, cf. e.g. Rojas (1996), Waszczyszyn (1999). In the simplest learning method CBP (Classical Back-Propagation) the weight vector is computed iteratively, applying gradients g_i of the cost function (5):

$$g_i = \frac{\partial E_{\mathscr{D}}}{\partial w_i}, \text{ for } i = 1, \ldots, W. \tag{15}$$

Radial Basis Function Neural Network (RBFN). The network RBFN, shown in Figure 4b, has a layer of Radial Basis Functions (RBFs) ϕ_k, which are used in

the following linear regression function (in fact the regression problem is linear in weights)

$$y(\mathbf{x};\mathbf{w}) = \sum_{k=0}^{K} w_k \phi_k(\mathbf{x}) = \phi_0(\mathbf{x})b + \sum_{k=1}^{K} w_k \phi_k(\mathbf{x}). \tag{16}$$

In Figure 4b weight w_0 is marked as a bias b and the corresponding RBF is $\phi_0(\mathbf{x}) = 1$.

The radial basis function, formulated in the feature space ϕ, has a general form

$$\phi_k(\mathbf{x}) = \phi(||\mathbf{x} - \boldsymbol{\mu}_k||), \tag{17}$$

There are many various forms of RBFs. The polynomial RBF, applied for the single input x and used in (7), can be written as:

$$\phi_k(x) = x^k. \tag{18}$$

From among many RBFs the most commonly applied is the exponential function:

$$\phi_k(\mathbf{x}) = \exp\Big\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\Big\}, \tag{19}$$

where the centres are assumed to be $\boldsymbol{\mu}_k$.

The function (19) is usually referred to as "Gaussian basis", although it does not fulfil the requirement of having a probabilistic interpretation. If we compare (19) with the formula of Gaussian density (95) we see that instead of $[(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}]^{-1}$ normalization parameter equals 1 in (19).

In a general form (19) the RBF has $D \times (D \times 3)/2$ independent parameters for $\mathbf{x} \in \mathscr{R}^D$. The simplest Gaussian RBF is obtained if the covariance matrix is assumed to be a symmetric and isotropic matrix $\boldsymbol{\Sigma}^{-1} = s^{-2}\mathbf{I}$, cf. discussion in Point (5.2). The corresponding Gaussian RBF is:

$$\phi_k(\mathbf{x}) = \exp\Big\{-\frac{1}{2s^2}(\mathbf{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} (\mathbf{x} - \boldsymbol{\mu}_k)\Big\}, \tag{20}$$

where s is sometimes called *spread parameter* which governs the spatial scale of Gaussian RBF.

The function (20) has $D+1$ parameters and in case $D = 1$, i.e. for the single input the formula (20) takes the form:

$$\phi_k(x) = \exp\Big\{-\frac{(x - c_k)}{2s^2}\Big\}, \tag{21}$$

where sometimes the mean is denoted as a *centre* c_k.

The regression function (16) is linear with respect to weights. So far for $K + 1 \leq N$ we can obtain the solution applying the linear regression method, fully

corresponding to the LS method. Obviously, this method can be used for fixed RBFs, i.e. for known means and spread parameters of the Gaussian RBFs.

A special case, called *interpolation RBFN*, see Haykin (1999), p.277, corresponds to placing centres of RBFs in the points of the input set, i.e.

$$\boldsymbol{\mu}_k \equiv \mathbf{c}_k = \mathbf{x}^k, \tag{22}$$

where for $W = K+1 \leq N$ the centres are selected using various optimization criteria.

In a case of the interpolation RBFN the following vector (one-column matrix) is formulated for the input pattern points:

$$\boldsymbol{\phi}_{(W\times 1)} = [\phi_0(\mathbf{x}^n), \phi_1(\mathbf{x}^n), \ldots, \phi_K(\mathbf{x}^n)]^T, \tag{23}$$

where $W = K+1 \leq N$. The design matrix is composed of N rows vectors $\boldsymbol{\phi}^T_{(N\times 1)}(\mathbf{x}^n)$ corresponding to the number of patterns $n = 1, \ldots, N$ and takes the following form:

$$\boldsymbol{\Phi}_{(N\times W)} = \begin{bmatrix} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \ldots & \phi_K(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \ldots & \phi_K(\mathbf{x}^2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}^N) & \phi_1(\mathbf{x}^N) & \ldots & \phi_K(\mathbf{x}^N) \end{bmatrix}, \tag{24}$$

where:

$$\phi_{nk} = \phi_k(\mathbf{x}^n) = \exp\Big\{-\frac{1}{2s^2}[t^k - y(\mathbf{x}^n;\mathbf{w})]^2\Big\}. \tag{25}$$

Substituting the approximation (16) and vector (22) into (5) the following form of Eq. (6) can be derived:

$$\boldsymbol{\Phi}\mathbf{w} = \mathbf{t}, \tag{26}$$

where $\boldsymbol{\Phi}$ is design matrix (24) and $\mathbf{t} = \{t^n\}_{n=1}^N$. The left-hand side multiplication of this equation by $\boldsymbol{\Phi}^{\mathrm{T}}$ enables us to obtain the solution:

$$\mathbf{w}_{\mathrm{LS}} = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \equiv \boldsymbol{\Phi}^{\dagger}\mathbf{t}, \tag{27}$$

where the Moore-Penrose pseudo-inverse matrix $\boldsymbol{\Phi}^{\dagger}$ is used and the subscript LS is added in order to mark that solution (27) fully corresponds to that obtained by the application of the Least Square method.

Solution (27) can be unstable so the minimizer with weight-decay (8) is suggested to be used. This leads to the following solution

$$\mathbf{w}_{\mathrm{PLS}} = [(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}) + \lambda\mathbf{I}]^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}. \tag{28}$$

The added subscript PLS corresponds to the Penalized LS method with regularization parameter $\lambda > 0$.

Despite using nonlinear Basis Functions (especially RBFs) the formulation supported on the LS method is called "linear-in-the-parameter **w**" formulation.

3 Some Problems of ANNs Design

3.1 Evaluation of regularization parameter λ

It was mentioned in Point 2.3 that the value of regularization parameter λ controls the complexity of regression models. The network RBFN can be treated as a representative model for illustrating the analysis of regression problems. Now the problem of an optimal value of λ_{opt} is analysed basing on the network error analysis.

Example 3.1, taken from Tipping (2004). This example continues Example 2.1. The same mathematical regression function $h(x) = \sin 2\pi x \in [0.0, 1.0]$ is applied but the randomly generated synthetic data set is extended to $N = 15$ patterns, marked in Figure 5 by black points. This set is called a training set. Additional 15 grain-points are also randomly selected and constitute a validation set.

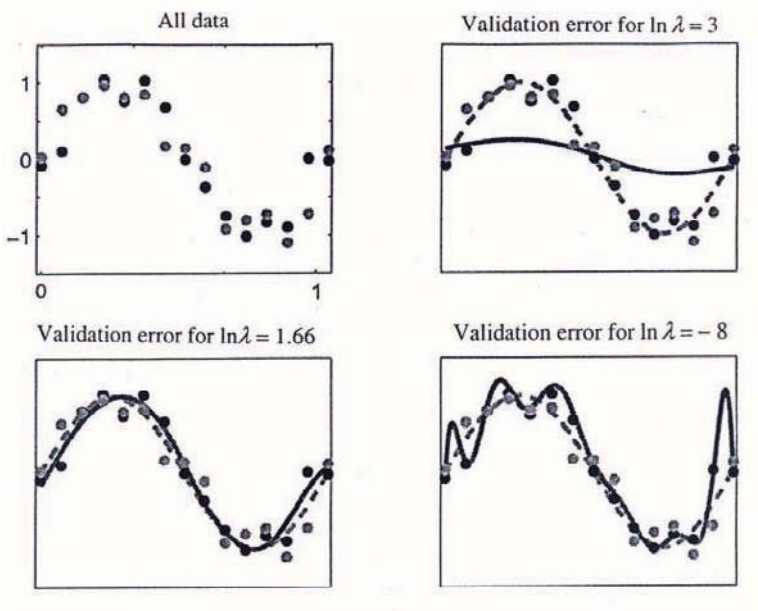


Figure 5. Mathematical sinusoidal regression function and randomly selected synthetic training and validation points. The regression function $y(x; \mathbf{w}, \lambda)$ was computed for $\ln \lambda = 3, 1.66, -8.0$

In Figure 6 normalised errors $E_{\mathscr{S}}(\lambda)$ are shown, where $\mathscr{S} = \mathscr{L}, \mathscr{V}$ are training (learning) and validation sets, respectively. The training was performed for $K = 14$ Gaussian RBFs (i.e. for $W = 15$ network parameters) with centres placed in the pattern points (x^n, y^n), randomly selected for the spread parameter $s = 0.2$. The training curve $E_{\mathscr{L}}(\lambda)$ is monotonically decreasing and for the log value $\ln\lambda = -\infty$ regularisation does not work.

The trained of RBFN with the weights $\mathbf{w}_{\mathrm{PLS}}(\lambda)$ was used for computation of the validation curve $E_{\mathscr{V}}(\lambda)$. The minimal value at this curve is reached for $\ln\lambda^{\mathscr{V}}_{\mathrm{opt}} \approx 1.66$ and it estimates the optimal value of the regression parameter $\lambda^{\mathscr{V}}_{\mathrm{opt}} \approx 5.3$.

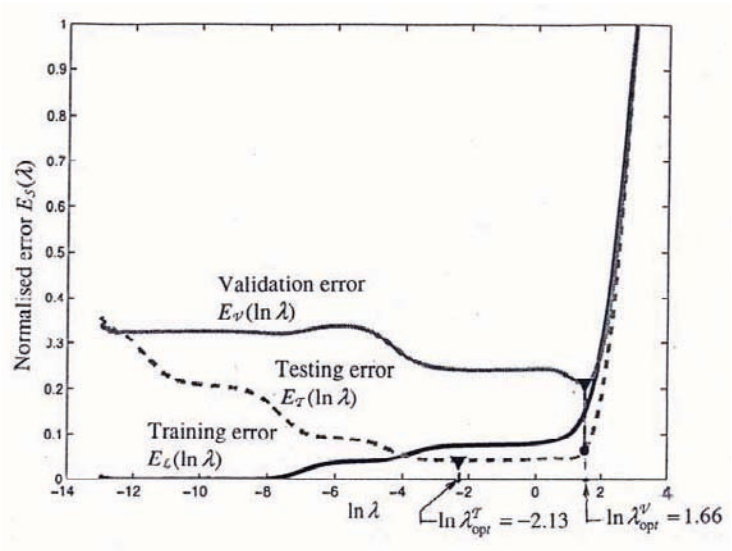


Figure 6. Error curves $E_{\mathscr{S}}(\lambda)$ and minimal value points at testing and validation curves $E_{\mathscr{T}}(\lambda)$ and $E_{\mathscr{V}}(\lambda)$, respectively

In Figure 6 the plot of the testing error is shown for the error measure $E_{\mathscr{T}}(\lambda) = ||y(x; \mathbf{w}_{\mathrm{PLS}}(\lambda^{\mathscr{T}}_{\mathrm{opt}}) - h(x)||$, computed for the mathematical regression function $h(x) = \sin 2\pi x$. The estimated optimal value $\ln\lambda^{\mathscr{T}}_{\mathrm{opt}} \approx -2.13$ gives the testing error circa $E_{\mathscr{T}}(\lambda^{\mathscr{T}}_{\mathrm{opt}}) \approx 0.04$, much lower than the validation error $E_{\mathscr{V}}(\lambda^{\mathscr{V}}_{\mathrm{opt}}) \approx 0.25$. In Figure 5 the corresponding validation fitting curves are shown, computed for selected values of regression parameter $\ln\lambda = 3, 1.66, -8.0$.

The described method of searching $\lambda^{\mathscr{V}}_{\mathrm{opt}}$ corresponds to the classical cross-validation method. The "true" value of $\lambda^{\mathscr{T}}_{\mathrm{opt}}$ is based on a known mathematical regression function. It is rather a luxurious situation and in computational practice only the search of $\lambda^{\mathscr{V}}_{\mathrm{opt}}$ is commonly carried out.

The estimated value of the regression parameter $\lambda^{\mathscr{V}}_{\mathrm{opt}}$ is strongly affected by a

randomly selected validation set of patterns. This question is not so important for large, statistically representative validation sets. The problem arises if the validation set is small or its selection from a known set of patterns significantly diminishes the size of the training set. That is why other error measures or approaches to designing of ANNs are worth attention. This question is investigated in Point 5.6.

3.2 Some remarks on cross-validation

The idea of designing neural networks on the base of validation set patterns has been discussed in many papers and books, cf. e.g. Masters (1993), Rojas (1996), Twomey and Smith (1997), Haykin (1999). The main point of this approach is that the validation set $\mathscr{V}$ is selected from a larger set of known patterns $\mathscr{P}$ and the remaining subset $\mathscr{L}$ is used as the training set, where $\mathscr{P} = \mathscr{L} \cup \mathscr{V}$, $\mathscr{L} \cap \mathscr{V} = \varnothing$, see Figure 7.

The sets $\mathscr{L}$ and $\mathscr{V}$ should be statistically comparable. A set of neural networks is learnt (trained) on the training set of patterns for various values of a parameter, representative for the NN model in question. This parameter usually corresponds to regularization parameter λ, the number of neurons H in FLNN or the number K of RBFs in the network RBFN, cf. Figures 4. Let us focus on H as the representative NN parameter. After the training is carried out, the validation curve $E(\mathscr{V};H)$ can be computed for fixed values of H.

On the base of the plot of validation curve we can evaluate a minimal value of validation error curve $E(\mathscr{V};H)$ and deduce a corresponding optimal value of the model parameter. Contrary to the training curve $E(\mathscr{L};H)$, which is *monotonically decreasing* for the increase of model complexity, the *validation curve* $E(\mathscr{V};H)$ *has a minimum* at H_{opt}.

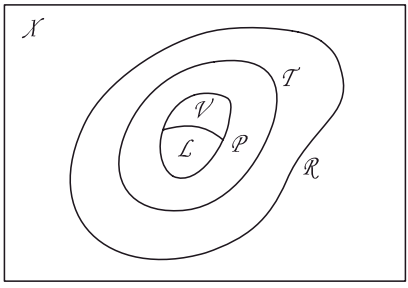


Figure 7. Sets of patterns in behaviour space $\mathscr{X}$

It is interesting that we can apply not only various model parameters but also different error measures to evaluate the training and validation curves. For instance, in a book by Bishop (1995), p. 11, two different measures *MSE* and *RMS*

(cf. definitions in Appendix A1) were applied for the training and validation errors, respectively. In Point 5.6 we apply another function, called the Bayesian marginal likelihood whose maximal value can be used as a criterion for evaluation of H_{opt}.

After cross-validation the network is ready for operation and we should test the trained network on a testing set of patterns $\mathscr{T}$, which should be independent of the set $\mathscr{P}$, see Figure 7. It is a luxurious situation if the testing set corresponds to the mathematical model, e.g. we know the mathematical regression curve $h(x)$ or if we could can formulate a noise-free set of patterns. Very often the set $\mathscr{P}$, if formulated via numerical simulations, gives so-called *pseudo-empirical data* and the testing set is taken from observations or physical experiments. The only requirement is that the set $\mathscr{P}$ should be included in a set $\mathscr{R}$ for which certain rules are valid. This set should, of course, be referred to the behaviour space $\mathscr{X}$, see Figure 7.

According to suggestions in some references, see e.g. Masters (1993), Twomey and Smith (1997), it is reasonable (especially for a small set $\mathscr{P}$) to retrain an optimal network on the full set of known patterns $\mathscr{P}$. After such a process, verification of the retrained NN on a testing set is invaluable.

All the above remarks take into account the *main goal of ANNs formulation*. In the regression problems the trained networks should well predict values of the network outputs in a larger space $\mathscr{R}$ in which certain rules obey so the testing process should approach us to the *main goal of NNs designing*, i.e. to have *a good prediction of trained NNs*.

In many applications we have only a known pattern set $\mathscr{P}$. If the set is large enough it can be split into S subsets of comparable size, see Figure 8. One of the subsets is used for training and the remaining data are used for validation. This process is executed S times. Such an approach was called the *multi-fold cross-validation* method, see Haykin (1999), pp. 217-218.

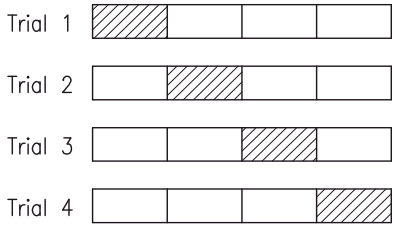


Figure 8. Selection of training and validation sets after division $S = 4$ times of a known set $\mathscr{P}$ into four subsets

In the often applied modification of multi-fold cross-validation a set of a fixed number of training patterns is randomly selected S times and trained on the complement to the known set $\mathscr{P}$. A quasi-optimal neural network then trained gives

the training error for a predicted regression function $y^{(s)}(x)$, close to the average error computed for the committee of S networks.

In case the known data set $\mathscr{P}$ is small and, particularly, scarce it may be appropriate to apply *leave-one-out technique* in which one after the other patterns are used for validation and the remaining patterns of the set $\mathscr{P}$ complete the training sets.

3.3 Design of ANNs with variable number of RBFs neurons

The ratio of K/N, where: K – the number of RBFs, N – the number of training patterns, influences the accuracy of network prediction at a fixed value of spread parameter s, assumed in all the RBFs. The adaptive search of K and the application of different values s is included by means of special procedures taken for instance from the manual by Demuth and Beale (1998), called Neural Network Toolbox for Use with MATLAB.

There are also extensively developed general types of RBFNs with the optimal placement of RBF centres out of the data points, as well as searching of optimal number of RBFs and their parameters, see Haykin (1999), Ch.5. Other approaches are related to the application of Support Vector Machines (SVM) or Relevance Vector Machines (RVM), cf. e.g. Haykin (1999)) and Bishop (2006). The main idea of these approaches is discussed in Point 5.9.

The situation with the Feed-forward Layered Neural Networks (FLNNs) is quite similar. We commonly apply the cross-validation procedure, recently also with weight-decay regularization for controlling the over-fitting phenomena. In case of one hidden layer the number of hidden neurons H can be used as a representative model parameter. The cross-validation method is then commonly applied and a minimum of the validation error $E_{\mathscr{V}}$ is used as a criterion of searching an optimal value of the neuron number $H^{\mathscr{V}}_{\text{opt}}$.

The application of the Bayesian framework enables us to extend our possibilities to find the best models of neural networks. This question is discussed at length in Point 5.6.

3.4 Data pre-processing and reduction of input space dimensionality

Scaling of input data. For most applications, it is necessary first to transform the data into some new representation before network training. Scaling or normalization is usually applied to have dimensionless data, often transformed to certain ranges. Very often the selection of transformation is enforced by applied procedures corresponding to physical interpretations, e.g. transformation of data from time to spectral spaces.

Let us focus on standard transformations. Besides rescaling and normalization onto the ranges $[0.0, 1.0]$ or $[-1.0, 1.0]$ the input vector components x_j can be

transformed by means of statistical parameters of the data set:

$$\hat{x}_j^n = \frac{x_j^n - \bar{x}_j}{\sigma_j^2}, \tag{29}$$

where: $n = 1,\ldots,N$ – superscript which labels the patterns, $j = 1,\ldots,D$ – subscript of components in the input space $\mathscr{R}^D$; $\bar{x}_j$, σ_j^2 – mean and standard deviation of the data set, defined by the following formulae:

$$\bar{x}_j = \frac{1}{N}\sum_{n=1}^{N} x_j^n, \tag{30}$$

$$\sigma_j^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_j^n - \bar{x}_j)^2. \tag{31}$$

The transformed variables $\hat{x}_j^n$ have zero mean and unit variance σ_j^2 (or unit standard deviation σ_j). In case of regression problems it is often appropriate to apply transformation (29) to both input variables x_j and computed output or target variables y_j and t_j.

In practice input normalization ensures that all of the input variables are of order of unity. In this case we expect that the weights of the first hidden layer can also be of order of unity. Sometimes the normalization range $[0.1,0.9]$ is applied if the binary sigmoid is used as the activation functions in the network layers. The use of this range might be important if such an activation function were applied in the output neurons. That is not our case, since in this Chapter only linear, identity outputs are used.

In case of RBFs with spherically symmetric basis functions, it is particularly important to normalize the input vectors so that they span similar ranges. A simple linear rescaling (29) treats all the vector components as independent. A more sophisticated linear rescaling is known in literature as whitening, see Bishop (1995), pp. 299- 300.

For convenience, let us use the vector of input variables $\mathbf{x} = \{x_1,\ldots,x_D\}$ and define the mean vector $\bar{\mathbf{x}}$ and covariance matrix $\mathbf{S}$:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^n, \tag{32}$$

$$\mathbf{S} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^{\mathrm{T}}. \tag{33}$$

The eigenvalue equation for the covariance matrix is in the form

$$\mathbf{S}\mathbf{q}_j = \lambda_j \mathbf{q}_j, \tag{34}$$

which enables us to obtain the transformed input vector:

$$\hat{\mathbf{x}}^n = |\mathbf{\Lambda}|^{-1/2}\mathbf{Q}^{\mathrm{T}}(\mathbf{x}^n - \bar{\mathbf{x}})^{\mathrm{T}}, \tag{35}$$

where the eigenvector matrix $\mathbf{Q}$ and determinant of the diagonal eigenvalue matrix $\mathbf{\Lambda}$ are defined as:

$$\mathbf{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_D\}, \tag{36}$$

$$|\mathbf{\Lambda}| = \prod_{j=1}^{D} \lambda_j. \tag{37}$$

Figure 9 shows that the original distribution of principal eigenvectors is transformed to the whitened distribution since the covariance matrix (33) becomes the unit matrix.

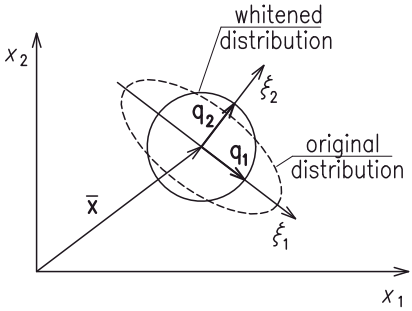


Figure 9. Transformation of input data set $\{\mathbf{x}^n\}$ to whitened distribution

The discussed transformation of input variables is the base of the PCA (*Principal Component Analysis*) method, which is commonly used for reduction of the input space dimensionality. PCA is presented below.

Principal Component Analysis. The *Principal Component Analysis* (PCA) is one of the commonly applied methods of reduction of input space dimension. PCA enables us to avoid loss of relevant information which could accompany the dimensionality reduction.

PCA is a linear transformation (projection) of the input data $\mathbf{x}^n = \{x_1^n, \ldots, x_D^n\} \in \mathscr{R}^D$ onto a lower dimensional space $\boldsymbol{\xi}^n = \{\xi_1^n, \ldots, \xi_K^n\} \in \mathscr{R}^K$, where $K < D$. The goal of transformation is to maximize the variance of projected data or, equivalently, to minimize the sum-of-squares of the projection errors, cf. Figure 10a taken from Bishop (2006).

The principal components are measured along the axis ξ_j directed by the basis vectors $\mathbf{q}_j$, cf. Figure 10b, which fulfill the criterion of orthonormality:

$$\mathbf{q}_i\mathbf{q}_j = \delta_{ij}, \tag{38}$$

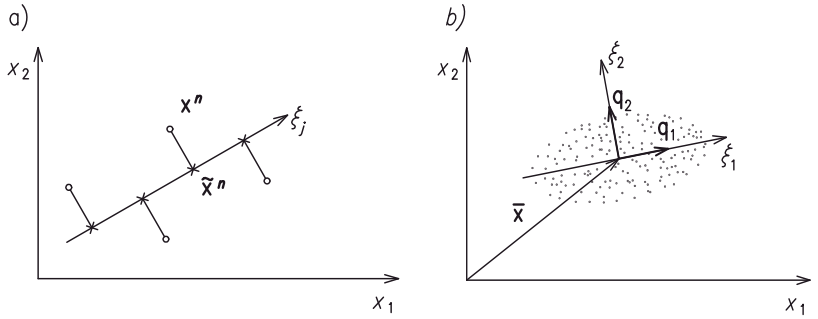


Figure 10. a) Projection of points $\mathbf{x}^n$ onto principal line ξ_j, b) Translation and rotation of principal lines ξ_1 and ξ_2

where: δ_{ij} – Kronecker delta.

Let us assume that the set of basis vectors is complete, that is

$$\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_D], \tag{39}$$

so the input points can be exactly expressed by a linear combination of the basis vectors

$$\mathbf{x}^n = \sum_{j=1}^{D} \alpha_j^n \mathbf{q}_j \tag{40}$$

where:

$$\alpha_j^n = (\mathbf{x}^n)^{\mathrm{T}} \mathbf{q}_j. \tag{41}$$

Let us approximate the input data involving a restricted number $K < D$ of variables corresponding to projection onto a lower-dimensional subspace, spanned on the K first basis vectors $\mathbf{q}_j$ for $j = 1, \ldots, K$. Thus, the approximate vector can be written in the form

$$\tilde{\mathbf{x}}^n = \sum_{j=1}^{K} z_j^n \mathbf{q}_j + \sum_{j=K+1}^{D} b_j \mathbf{q}_j. \tag{42}$$

Let us introduce the mean-square-error function

$$J = \frac{1}{N} \sum_{n=1}^{N} ||\mathbf{x}^n - \tilde{\mathbf{x}}^n||^2. \tag{43}$$

The minimization of J with respect to coefficients z_j^n and b_j gives

$$z_j^n = (\mathbf{x}^n)^{\mathrm{T}} \mathbf{q}_j, \quad b_j = \bar{\mathbf{x}}^{\mathrm{T}} \mathbf{q}_j. \tag{44}$$

The obtained coefficients (44) lead to zero value of the first part of sum (42), i.e. for $j = 1, \ldots, K$ and the error function is reduced to the formula

$$J = \frac{1}{N} \sum_{n=1}^{N} \sum_{j=K+1}^{D} \{(\mathbf{x}^n)^{\mathrm{T}} \mathbf{q}_j - \bar{\mathbf{x}}^{\mathrm{T}} \mathbf{q}_j\}^2 \equiv \sum_{j=K+1}^{D} \mathbf{q}_j^{\mathrm{T}} \mathbf{S} \mathbf{q}_j, \tag{45}$$

where $\mathbf{S}$ is the covariance matrix and $\bar{\mathbf{x}}$ is the mean of the data set

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^{\mathrm{T}}, \quad \bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}^n. \tag{46}$$

The general solution for the minimization of J for arbitrary D and $K < D$ is based on the eigen-analysis of equation (34). Since the covariance matrix is symmetric and positive defined, there are D positive eigenvalues which we put in order

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D. \tag{47}$$

Applying the definition of the data set variance

$$\sigma_j^2 = \mathbb{E}[A_j^2] \equiv \mathbb{E}[(\mathbf{q}_j^{\mathrm{T}} \mathbf{X})(\mathbf{X}^{\mathrm{T}} \mathbf{q}_j)] = \mathbf{q}_j^{\mathrm{T}} \mathbb{E}[\mathbf{X}\mathbf{X}^{\mathrm{T}}] \mathbf{q}_j \equiv \mathbf{q}_j^{\mathrm{T}} \mathbf{S} \mathbf{q}_j, \tag{48}$$

where: $\mathbf{X} = (\mathbf{x} - \bar{\mathbf{x}})$, and substituting (34) into (48) we obtain

$$\sigma_j^2 = \mathbf{q}_j^{\mathrm{T}} \lambda_j \mathbf{q}_j = \lambda_j \mathbf{q}_j^{\mathrm{T}} \mathbf{q}_j \equiv \lambda_j. \tag{49}$$

Substituting (48) to (45) we can conclude that the PCA approximation error is

$$J = \sum_{j=K+1}^{D} \lambda_j. \tag{50}$$

Now we can formulate the following PCA algorithm:

1^0 Formulation of the covariance matrix $\mathbf{S}$ for N given data points;

2^0 Computation of the eigenpairs $(\lambda_j, \mathbf{q}_j)$ for the matrix $\mathbf{S}$;

3^0 Computation of the relative values of the eigenvalues

$$m_j = \frac{\lambda_j}{\lambda_{\mathrm{T}}}, \text{ where } \lambda_{\mathrm{T}} = \sum_{j=1}^{K} \lambda_j, \tag{51}$$

and estimation of a value K such that

$$\sum_{j=1}^{K} m_j \leq \text{adm error} \leq \sum_{j=K+1}^{D} m_j; \tag{52}$$

4^0 The formulated approximation enables us to compute the condensed data (PC vectors)

$$\boldsymbol{\xi}^n = \mathbf{Q}^{\mathrm{T}}\mathbf{x}^n \equiv \sum_{j=1}^{K} x_j^n \mathbf{q}_j; \tag{53}$$

5^0 PC vector can also serve for reconstruction of the original data

$$\tilde{\mathbf{x}}^n \approx \tilde{\mathbf{Q}}^{\mathrm{T}}\boldsymbol{\xi}^n = \sum_{j=1}^{K} \xi_j^n \mathbf{q}_j, \tag{54}$$

where: $\tilde{\mathbf{Q}}$ – incomplete basis matrix

$$\tilde{\mathbf{Q}}_{(D\times K)} = \{\mathbf{q}_1, \ldots, \mathbf{q}_K\}. \tag{55}$$

4 Applications of Kalman Filtering to Learning of ANNs

4.1 Sequential data and Kalman filtering

So far we have focused on sets of events that were assumed to be *independent and identically distributed* (i.i.d.). However, there are many observed phenomena for which the data indicate correlations between events that are close in the sequence. Such sequential data can be easily put in order if the events are related to a monotonically increasing parameter. The physical time t is frequently used in dynamics of systems but also any other pseudo-time parameters τ of monotonically increasing values can be used in the sequential data $y(\tau)$.

In what follows we focus on discrete pseudo-time $\tau \equiv k = 1, 2, \ldots, K$ which can be used to put in order the events in the set of sequential data

$$\{y_1, y_2, \ldots, y_{k-1}, y_k, y_{k+1}, \ldots, y_K\}, \tag{56}$$

which are correlated to each other. This feature can be expressed by the conditional probability

$$\begin{aligned} p(y_1, y_2, \ldots, y_K) &= \prod_{k=1}^{K} p(y_k | y_1, \ldots, y_{k-1}) \\ &= p(y_1)p(y_2|y_1)\cdots p(y_K|y_1, \ldots, y_{K-1}). \end{aligned} \tag{57}$$

In many observations it is evident that the correlations between the current event and the previous ones are relaxing, which is the basic assumption of Markov models. In the theory of *Markov chains* correlations of the current and several previous m events are discussed. This can be related to Markov chains of m order. Let

us illustrate this definition on the examples of the first and second order Markov chains, see also Figure 11:

$$m=1:\ p(y_1,y_2,\ldots,y_K)=p(y_1)\prod_{k=2}^{K}p(y_k|y_{k-1}), \tag{58}$$

$$m=2:\ p(y_1,y_2,\ldots,y_K)=p(y_1)p(y_2|y_1)\prod_{k=3}^{K}p(y_k|y_{k-1},y_{k-2}). \tag{59}$$

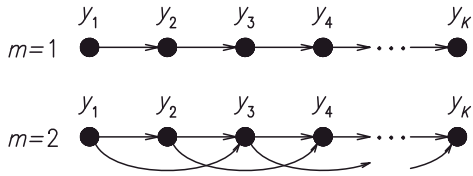


Figure 11. First and second order Markov chains

The idea to preserve only several time-delay terms is explored in the time series. The 1-st order Markov chain is assumed to formulate the Kalman filter. The second assumption concerns the introduction of a state variable $\mathbf{x}_k$ and formulation of a model in the state space.

Let us assume that the state variable fulfils the assumption of the first order Markov chain and the observable variables $\mathbf{y}_k$ are i.i.d. but they depend on state variables $\mathbf{x}_k$ according to the following probabilistic formula, see Figure 12:

$$p(\mathbf{x}_1,\ldots,\mathbf{x}_K,\mathbf{y}_1,\ldots,\mathbf{y}_K)=p(\mathbf{x}_1)\prod_{k=2}^{K-1}p(\mathbf{x}_{k+1}|\mathbf{x}_k)\prod_{k=1}^{K}p(\mathbf{y}_k|\mathbf{y}_{k-1}). \tag{60}$$

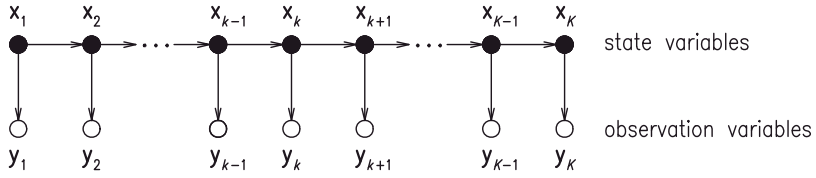


Figure 12. Sequences of state and observation variables

The model defined by (60) and Figure 12 was used in the formulation of the Kalman filter which explores the recurrent formulae for computing vectors as mappings $\mathbf{x}_k \rightarrow \mathbf{x}_{k+1}$ and $\mathbf{x}_k \rightarrow \mathbf{y}_k$. The state and observation vectors $\mathbf{x}_k$ and $\mathbf{y}_k$ are assumed to be random variables.

The approach described above is a base for formulating the linear Kalman filter which is widely explored in the stochastic analysis of discrete linear dynamic systems, see e.g., Haykin (1999).

There are many problems with the formulation of nonlinear Kalman filters, cf. e.g. Haykin (1999, 2001), Korbicz et al. (1994). Just in this field the application of neural networks has permitted overcoming some basic difficulties with formulating so-called extended Kalman filters for the analysis of nonlinear discrete dynamic systems, see Haykin (1999, 2001).

4.2 Kalman filtering and KF learning algorithms

The Kalman filtering algorithm is based on the modified first order Markov chain model (60). In this model the NN weight vector $\mathbf{w}$ is assumed to be the state vector. This leads to two following stochastic equations:

1) *process equation*

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \boldsymbol{\omega}_k, \tag{61}$$

2) *observation* (measurement) *equation*

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{w}_k) + \mathbf{e}_k, \tag{62}$$

where: $\mathbf{h}_k$ – nonlinear regression function, $\mathbf{x}_k$ – input vector, $\boldsymbol{\omega}_k$, $\mathbf{e}_k$ – process and measurement noises of zero mean and covariance matrices

$$\mathbb{E}[\boldsymbol{\omega}_k] = \mathbb{E}[\mathbf{e}_k] = 0, \tag{63}$$

$$\mathbb{E}[\boldsymbol{\omega}_k \boldsymbol{\omega}_l] = \boldsymbol{\delta}_{kl}\mathbf{Q}_k, \quad \mathbb{E}[\mathbf{e}_k \mathbf{e}_l] = \boldsymbol{\delta}_{kl}\mathbf{R}_k. \tag{64}$$

GEKF algorithm. The *Global Extended Kalman Filter* is based on Eqs (61, 62) and the application of the following recursive formulae, see Haykin (2001), p. 29:

$$\begin{aligned} \mathbf{A}_k &= [\mathbf{R}_k + \mathbf{H}_k^{\mathrm{T}}\mathbf{P}_k\mathbf{H}_k]^{-1} \\ \mathbf{K}_k &= \mathbf{P}_k\mathbf{H}_k\mathbf{A}_k \\ \hat{\mathbf{w}}_{k+1} &= \hat{\mathbf{w}}_k + \mathbf{K}_k\mathbf{e}_k \\ \mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{K}_k\mathbf{H}_k^{\mathrm{T}}\mathbf{P}_k + \mathbf{Q}_k, \end{aligned} \tag{65}$$

where: $\mathbf{A}_k$ – global scaling matrix; $\mathbf{K}_k$ – Kalman gain matrix; $\hat{\mathbf{w}}_{k+1}$, $\hat{\mathbf{w}}_k$ – estimate of weights of the system at updated steps $k+1$ and k; $\mathbf{P}_k$ – error covariance matrix; $\mathbf{H}_k$ – measurement matrix; $\mathbf{e}_k$ – error vector; $\mathbf{R}_k$, $\mathbf{Q}_k$ – measurement and covariance noise matrices.

The names used in the acronym GEFK correspond to the form of Eqs (65). The algorithm is called *General* since it is applied to the whole dynamic system governed by Eqs (65). The name *Extended* is used since the commonly used Kalman

filters are linear but in (65) the measurement matrix $\mathbf{H}_k$ is nonlinear, i.e. more precisely, it will be linearized at each step k.

The error vector $\mathbf{e}_k$ is defined as the difference of the target and computed output vectors $\mathbf{t}_k$ and $\hat{\mathbf{y}}_k$ for the k-th presentation

$$\mathbf{e}_k = \mathbf{t}_k - \hat{\mathbf{y}}_k. \tag{66}$$

In case of a single output network the components of the target and computed vectors correspond to N variables at each observation step k

$$\mathbf{t}_k = \{t_k^n\}_{n=1}^N, \quad \hat{\mathbf{y}}_k = \{\hat{y}_k^n\}_{n=1}^N. \tag{67}$$

The algorithm attempts to find weight values that minimize a cost function, e.g. the Mean Square Error, cf. definition (A1):

$$MSE = \frac{1}{NK} \sum_{k=1}^{K} \mathbf{e}_k^{\mathrm{T}} \mathbf{e}_k, \tag{68}$$

The measurement matrix $\mathbf{H}_k$ is composed of derivatives of the network outputs with respect to all trainable weight parameters

$$\mathbf{H}_k = \frac{\partial \mathbf{h}_k(\mathbf{x}_k, \mathbf{w}_k)}{\partial \mathbf{w}} |_{\mathbf{w}=\hat{\mathbf{w}}_k}. \tag{69}$$

The derivative matrix is obtained by separate back-propagation for each component of the output vector $\hat{\mathbf{y}}_k$.

In the algorithm the key role is played by the *covariance matrix* $\mathbf{P}_k$

$$\mathbf{P}_k = \mathbb{E}[(\mathbf{w}_k - \hat{\mathbf{w}}_k)(\mathbf{w}_k - \hat{\mathbf{w}}_k)^{\mathrm{T}}]. \tag{70}$$

In order to start with the recursion the initial estimate $\hat{\mathbf{w}}_0$ and the covariance matrix $\mathbf{P}_0$ have to be assumed

$$\hat{\mathbf{w}}_0 = \mathbb{E}[(\mathbf{w}_0], \quad \mathbf{P}_0 = \mathbb{E}[(\mathbf{w}_0 - \hat{\mathbf{w}}_0)(\mathbf{w}_0 - \hat{\mathbf{w}}_0)^{\mathrm{T}}]. \tag{71}$$

The selection of corresponding initial values was discussed at length by Haykin (2001), pp. 32-33.

The algorithm GEKF, briefly discussed above, can be applied for training of Feed-forward Layered Neural Networks (FLNNs), Figure 13a. The network has two hidden layers and a linear output layer. The node activations for the l-th layer are marked as signals $\mathbf{v}_k^l$. An additional input, called auto-regression input $\mathbf{y}_{k-1}$, corresponds to the time delay. This input can accelerate the training process of the network in the analysis of some engineering problems.

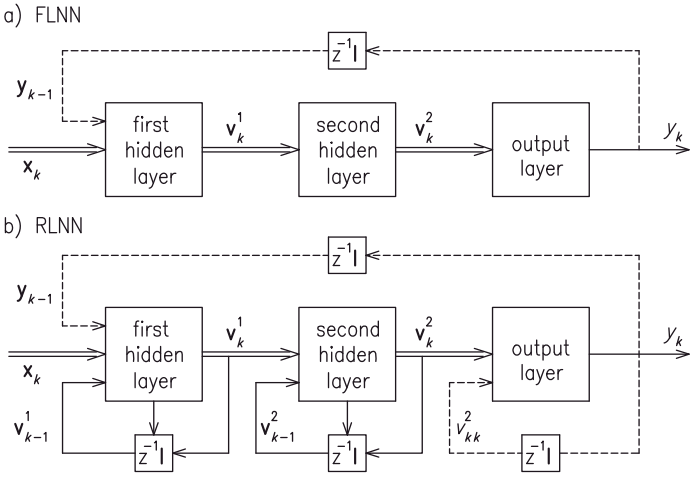


Figure 13. Layered Neural Network (LNN) with autoregressive input $\mathbf{y}_{k-1}$, a) Feed-forward LNN (FLNN), b) Recurrent LNN (RLNN)

The *noise vectors* ω_k and $\mathbf{e}_k$ stabilize the numerical procedures applied in training of the neural network. From among different possibilities of the matrices $\mathbf{R}_k$ and $\mathbf{Q}_k$ modelling the noises, a simple approach is related to assuming the diagonal matrices and making their parameters depending on the epoch number s

$$\mathbf{R}_k(s) = \alpha \exp(-s/\beta)\mathbf{I}, \quad \mathbf{Q}_k(s) = \alpha' \exp(-s/\beta')\mathbf{I}, \tag{72}$$

where: $\alpha, \alpha', \beta, \beta'$ – parameters of values estimated on the base of numerical experiments. During the training process, the stabilization activity of matrices (72) is great at the beginning of the training and then diminishes for the increase of s values.

Due to neural networks it is possible to extend the Kalman filtering application in the analysis of nonlinear problems. The corresponding algorithms are called Extended KF, see Haykin (1999, 2001). The algorithm GEKF is based on the nonlinear behaviour of Eq. (62) in which the nonlinear observation function $\mathbf{h}_k(\mathbf{x}_k, \mathbf{w}_k)$ is used. In the recursion formulae (65), the measurement matrix is used as an approximation updated at each step k.

DEKF algorithm. In order to diminish computation complexity and storage requirements a decoupling of formulae (65) is carried out, see Haykin (2001), pp. 33-35. The decoupling concerns the introduction of g weight groups. In case $g = W$, where W is the number of the network weights full decoupling takes place. Decoupling can also be referred to each neuron (node-decoupling) or to a layer (layer-decoupling). In such a way the Decoupled Extended Kalman Filter (DEKF) is formulated.

DEKF algorithm is related to the following recursive formulae, which distinguish formulae (65) from GEKF by adding the subscript j to the weights of individual groups and express the global vectors and matrices via concatenation of individual quantities. The DEKF algorithm can be written in the following recursive form:

$$\begin{aligned}
\mathbf{A}_k &= [\mathbf{R}_k + \sum_{j=1}^{g} (\mathbf{H}_k^j)^{\mathrm{T}} \mathbf{P}_k^j \mathbf{H}_k^j]^{-1} \\
\mathbf{K}_k^j &= \mathbf{P}_k^j \mathbf{H}_k^j \mathbf{A}_k \\
\hat{\mathbf{w}}_{k+1}^j &= \hat{\mathbf{w}}_k^j + \mathbf{K}_k^j \mathbf{e}_k \\
\mathbf{P}_{k+1}^j &= \mathbf{P}_k^j - \mathbf{K}_k^j (\mathbf{H}_k^j)^{\mathrm{T}} \mathbf{P}_k^j + \mathbf{Q}_k^j.
\end{aligned} \tag{73}$$

RLNN network. The Kalman filtering can also be applied to learning of the Recurrent Layered Neural Network (RLNN) shown in Figure 13b.

RLNN is a modification of the Elman network, see Pham and Liu (1995). RLNN is only internally recurrent since the time-delay of the vector of potential v_{k-1}^l is applied as the input to the layer l. These additional internal inputs are taken into account in the process and observation equations:

$$\{\mathbf{w}_{k+1}, \mathbf{v}_k\} = \{\mathbf{w}_k, \mathbf{v}_{k-1}\} + \boldsymbol{\omega}_k, \tag{74}$$

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{w}_k, \mathbf{v}_{k-1}) + \mathbf{e}_k. \tag{75}$$

Obviously, also in RLNN the auto-progressive input $\hat{\mathbf{y}}_{k-1}$ can be applied but in such a case the internal recursive input $\mathbf{v}_{k-1}^3 = \mathbf{y}_{k-1}$ is not introduced.

Example 4.1, taken from Lou and Perez (1996). The mathematical model is given by the curve $h(x) \equiv t(x)$ of the following equation

$$t(x) \equiv h(x) = 0.1 \exp(x + \sin(25x)) + 0.9 \sin(x + 10 \exp(x/2)), \tag{76}$$

with the input variable $x \in [0,1]$. This interval was divided into 100 equal length subintervals with $K = 101$ points corresponding to the ends of subintervals. Data from these points were assumed to be patterns of the training set $\{x_k, t_k\}_{k=1}^{K=101}$.

The standard network with one hidden layer FLNN: 1-17-1, see Figure 14, was proposed to compute the physical type curve $y(x; \mathbf{w})$ fitting well the points (x_k, t_k). The bipolar sigmoid was used in the hidden neurons defined in (12). The output is linear with the identity activation function (11).

The network shown has $NN = 7 + 1 = 8$ neurons and the number of weight parameters (synaptic weights and biases) equal $W = 2 \times 17 \times 1 + (17 + 1) = 52$. Initial values of the weights were randomly selected for $k = 0$ from the range $[-0.5, 0.5]$. Two learning methods were used:

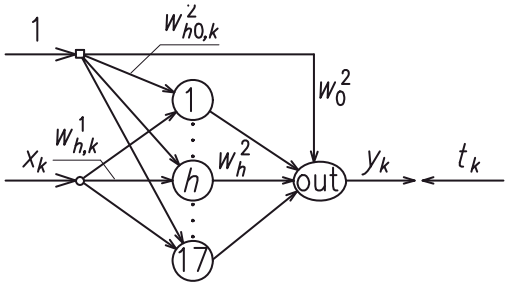


Figure 14. Neural network used for analysis of Example 4.1

1) classical method BP+M, i.e. back-propagation + momentum term, see Waszczyszyn (1999),
2) DEKF algorithm.

In case of BP+M application the learning rate $\eta = 0.005$ and momentum coefficient $\alpha = 0.5$ were used. The iterations were poorly convergent and after $S = 5000$ epochs the training error was $MSE(S) \approx 0.011$.

The DEKF algorithm was applied to the network FLNN shown in Figure 13a, without using autoregressive input y_{k-1}. The decoupling was performed with respect to $g = 8$ groups corresponding to the network neurons. The artificial noise was introduced according to formulae (72) with the parameters as written below:

$$R_k = 10\exp(-(s-1)/50), \quad \mathbf{Q}_k^j = 0.01\exp(-(s-1)/50)\mathbf{I}_{8\times 8}. \tag{77}$$

Training with the Kalman filtering was continued up to $S = 216$ epochs when the error $MSE(S) \approx 0.001$. In Figure 15 fitting of simulated curve $y(x;\mathbf{w})$ to the mathematical curve $h(x)$ is shown. It is clear that the application of DEKF algorithm gives the simulated curve much more fitting to $h(x)$ than the classical algorithm BP+M.

The Kalman filters are commonly applied in the analysis of many engineering problems related to linear dynamic systems (especially, in the control theory and pattern recognition), see Korbicz et al. (1994). Kalman filtering was used in networks applied to the analysis of structural dynamics problems, see Sato and Sato (1997). Krok and Waszczyszyn (2007) applied DEKF and RDEFK algorithms in the simulation of so-called Response Spectra from the paraseismic excitations, made similar applications. These applications were developed in Krok's PhD. dissertation, see Krok (2007). In the dissertation, neural networks with Kalman filtering were also applied to the analysis of hysteresis loops in mechanics of structures and materials, see Krok (2006). Two examples of the Kalman Filtering in SNNs are discussed in Points 6.1 and 6.2.

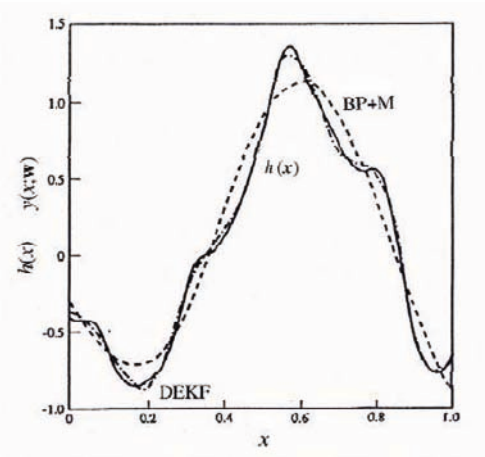


Figure 15. Fitting of curves $y(x;\mathbf{w})$, trained by algorithms DEKF and BP+M, to mathematical curve $h(x)$

It is worth mentioning that the Kalman filter algorithms can also be formulated on the base of Bayesian methods, as discussed in books by Haykin (2001) and Bishop (2006).

5 Bayesian Neural Networks

5.1 Bayesian vs. Standard Neural Networks

There are several reasons why the Bayesian Neural Networks (BNNs) are in the centre of attention of many researchers and engineers. BNNs have several features which distinguish them from the Standard Neural Networks (SNNs):

i) *BNNs* are *probabilistic networks*, vs. *SNNs* which are *deterministic*. That means that in BNNs random variables are used, and in the analysis not only means are searched (like in SNNs) but also the probability distribution of the used variables and parameters.

ii) BNN on the basis of the *Bayes' theorem* in which conditional probabilities are used for the inference of 'a posteriori' probability distribution (pd) on the base of known or earlier computed 'prior' pd.

iii) The Bayesian inference needs the computation of integrals over all the system parameters, called in short *marginalization of variables*. This creates the *marginalization paradigm* of BNNs, vs. the SSN *error minimization paradigm*.

iv) *Regularization* is introduced into BPN models in order to control the overfitting phenomena. This is not a common approach in SNNs since regularization or corresponding penalty functions are not usually introduced into formulated deterministic models.

v) *BNNs contain* in fact *SNNs* since the *maximum likelihood method*, leading to "not truly" BNNs, fully corresponds to the SNN paradigm of error minimization.

vi) BNNs seem to be better *mathematically founded* than SNNs in which many heuristic types of knowledge are introduced.

vii) *BNNs better fit the physical reality* of many problems analysed in science and technology (including, obviously, civil or mechanical engineering) due to the probabilistic character of those networks.

This Section is based mainly on Bishop's book, Bishop (2006) and Tipping's review, Tipping (2004) but we also return to an outstanding review by MacKay (1992) and contributions by other authors. In Section 5 the main ideas of Bishop's approach are presented, illustrated by simple examples. The attention is focused on various Bayesian computational methods suitable in the analysis of regression problems. New trends in BNNs are also discussed in short at the end of the Section.

5.2 Some basics from the probability theory

In order to understand problems discussed in this Section only some basics from the theory of probability were selected from the book by Bishop (2006), Sections 1, 2 and Appendix B.

Probability and Bayes' theorem. Let us consider two *random variables* X and Y and the *probabilities* that X will take value x_i and Y will take value y_j:

$$p(X = x_i), \quad p(Y = y_j). \tag{78}$$

The *joint probability*, written in the form:

$$p(X = x_i, Y = y_j) \tag{79}$$

is used for the case both variables are independent of each other.

The case of the probability of Y conditioned by X (this is verbalized as "the probability of Y given X") is defined as *conditional probability* $p(Y|X)$. In the same way we define probability $p(X|Y)$ of X given Y.

The joint probability $p(X,Y)$ can be expressed by conditional probabilities by means of the *product rule*:

$$p(X,Y) = p(Y|X)p(X) \tag{80}$$

$$= p(X|Y)p(Y), \tag{81}$$

where: $p(X)$ and $p(Y)$ – probabilities of X or Y, sometimes called *marginal probabilities*.

The second fundamental rule is the *sum rule*, which is written below with summing over $Y = y_j$:

$$p(X = x_i) = \sum_Y p(X,Y) \equiv \sum_{j=1}^{J} p(X = x_i, Y = y_j). \tag{82}$$

The two expressions (80) and (81) must be equal so the following *Bayes' theorem* can be derived:

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}. \tag{83}$$

The theorem can be expressed by means of the following *verbalization*:

$$posterior = \frac{likelihood \times prior}{evidence}, \tag{84}$$

where the used words correspond to the terms commonly applied in the Bayesian analysis. The *evidence* plays the role of a factor which can be written by means of sum and product rules (80) and (81):

$$p(X) = \sum_Y p(X|Y)p(Y) \equiv \sum_{j=1}^{J} p(X|y_j)p(y_j). \tag{85}$$

Theorem (83) is applied to the *Bayesian inference* in which we start from a rough *'a priori'* estimation (prior) of the considered variable Y, for instance the probability density $p(Y)$. Formula (83) enables us to compute an *'a posteriori' estimation* (posterior) $p(Y|X)$ of Y given X, i.e. an additional knowledge is introduced related to X.

Probability densities. The probabilities discussed above are defined over discrete sets of events. We also examine probabilities with respect to continuous variables. *Probability density* (also called probability distribution - pd) is defined below over real variable which must satisfy two conditions:

$$p(x) \geq 0 \tag{86}$$

$$\int_{-\infty}^{\infty} p(x)dx = 1. \tag{87}$$

The pds can be related also to real variables x and y for which the product and sum rules take the form:

$$p(x,y) = p(y|x)p(x) \tag{88}$$

$$p(x) = \int p(x,y)dy, \tag{89}$$

where integral (89) is taken over the whole x axis.

Rules (88) and (89) can be easily generalized for multivariate variable $\mathbf{x} \in \mathscr{R}^D$:

$$p(\mathbf{x}) \geq 0 \tag{90}$$

$$\int_{\mathscr{R}^D} p(\mathbf{x},\mathbf{y})d\mathbf{y} = 1, \tag{91}$$

where $\mathscr{R}^D$ is a region (domain) of integration over variable $\mathbf{y}$.

The Bayes' theorem is valid also for real multivariate variables $\mathbf{x}$ and $\mathbf{y}$:

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{y})p(\mathbf{y})}{p(\mathbf{x})} \tag{92}$$

$$p(\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{y}, \tag{93}$$

where the region of integration $\mathscr{R}^D$ is usually omitted at the integral.

Gaussian probability distribution. In the Chapter we focus on the applications of the Gaussian probability density called *normal* pd. In case of single variable x the normal pd takes the form, drawn in Figure 16:

$$\mathscr{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}, \tag{94}$$

where: μ – *mean*, σ^2 – *variance* which can be related to other parameters, i.e. σ – *standard deviation*, $\beta = 1/\sigma^2$ – *precision*.

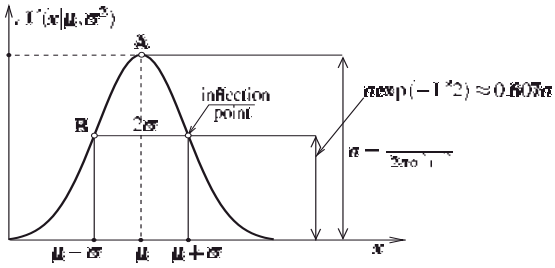


Figure 16. Gaussian (normal) probability distribution

The Gaussian pd shown in Figure 16 fulfils the requirements (90) and (91) due to the exponential function and the scaling parameter $a = (2\pi\sigma^2)^{-1/2}$ in formula (94).

In case of D-dimensional variable $\mathbf{x}$ the Gaussian pd takes the form:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big\}, \tag{95}$$

where: $\mathbf{x},\boldsymbol{\mu} \in \mathscr{R}^D$ – position and mean vectors, $\boldsymbol{\Sigma} = \mathbb{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}] \in \mathscr{R}^D \times \mathscr{R}^D$ – covariance matrix, $|\boldsymbol{\Sigma}|$ – determinant of $\boldsymbol{\Sigma}$.

The covariance matrix $\boldsymbol{\Sigma}$ is symmetric and has $D(D+1/2)/2$ independent parameters. The mean $\boldsymbol{\mu}$ has D parameters, so in the general multivariate case the Gaussian pd has $D(D+3)/2$ independent parameters.

It is sometimes convenient to examine a simplified form of a Gaussian pd in which the covariance matrix is diagonal with components

$$\Sigma_{ij} = \delta_{ij}\sigma_j^2. \tag{96}$$

The simplified matrix leads to $2D$ independent parameters of the Gaussian pd. Further simplification can be obtained by choosing $\sigma_j = \sigma$ for all j. This leads to the isotropic covariance matrix

$$\boldsymbol{\Sigma} = \sigma\mathbf{I}, \tag{97}$$

and the corresponding Gaussian pd has then $D+1$ independent parameters.

Data points drawn independently from the same distribution are said to be *independent and identically distributed* (often abbreviated to i.i.d.). In such a case, we can write the probability of data set $\{x^n\}_{i=1}^N$, given μ and σ^2, in the form:

$$p(\mathbf{t}|\mathbf{X},\mathbf{w},\sigma^2) = \prod_{n=1}^{N} \mathcal{N}(t^n|y(\mathbf{x}^n;\mathbf{w}),\sigma^2). \tag{98}$$

5.3 Bayesian inference

In Point 2.1 we discussed the regression problem and illustrated it on a case study of curve fitting. Following this approach, let us return to the physical model of regression assuming that the target variable t is given by a *deterministic curve* $y(x;\mathbf{w})$ with *additive Gaussian noise* ε

$$t = y(x;\mathbf{w}) + \varepsilon. \tag{99}$$

We can express our uncertainty over the value of t using a probability distribution, which is assumed to be a *Gaussian noise*. The noise is independent of observed patterns $(\mathbf{x}^n,t^n)$ and has the normal pd of zero mean and variance σ^2, i.e. $p(\varepsilon|\sigma^2) = \mathcal{N}(0,\sigma^2)$. This leads to pd for t defined by (99):

$$p(t|x,\mathbf{w},\sigma^2) = \mathcal{N}(t|y(x;\mathbf{w}),\sigma^2), \tag{100}$$

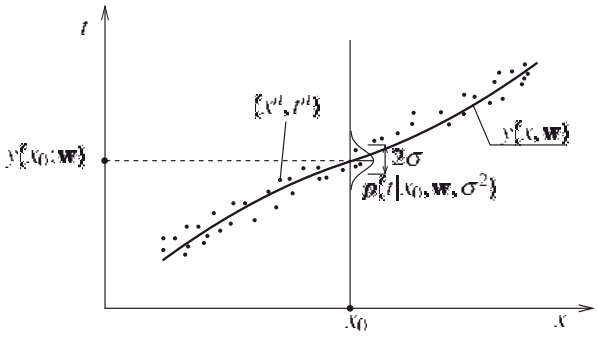


Figure 17. Observation points (x^n, t^n), simulated curve $y(x;\mathbf{w})$ and Gaussian conditional distribution $p(t|x_0, \mathbf{w}, \sigma^2)$

where the function $y(x;\mathbf{w})$ plays the role of the mean and σ^2 is the noise variance. In Figure 17 the observation points (x^n, t^n) are shown, numerically simulated with variance σ^2 at the curve $y(x;\mathbf{w})$ as well as Gaussian pd (100).

It was assumed above that the observation t^n is referred to multidimensional domain $\mathbf{x} \in \mathscr{R}^D$ (input space in case of SNNs) so that data set $\mathscr{D}$ is completed as pairs $(\mathbf{x}^n, t^n)$:

$$\mathscr{D} = \{\mathbf{x}^n, t^n\}_{n=1}^N \equiv \{\mathbf{X}, \mathbf{t}\}, \tag{101}$$

where the input and target subsets are:

$$\mathbf{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}, \quad \mathbf{t} = \{t^1, \ldots, t^N\}. \tag{102}$$

The main goal of regression analysis is to find a *scalar regression function* $y(\mathbf{x};\mathbf{w})$, probability distributions of the *weight parameters* vector $p(\mathbf{w})$ and the vector of target values $\mathbf{t}$. The analysis is based on *discrete data set* $\mathscr{D}$ in which we have only point observations $t^n(\mathbf{x}^n)$. This defines three basic quantities: 1) *data set* $\mathscr{D}$, 2) *set of parameters* $\boldsymbol{\theta} = \{\mathbf{w}, \alpha, \beta\}$, 3) *numerical model* which can be specified as a set of hypotheses $\{\mathscr{H}_l\}$ related to network architecture or computational algorithms.

The analysis is based on *Bayes' theorem* which reflects various relations between the conditional probabilities of the mentioned variables. A general form of Bayes' theorem is defined as formula (92).

In order to approach the Bayesian inference let us consider two simple problems called Examples 5.1 and 5.2.

Example 5.1. The Bayesian inference is applied to prediction of probability $p(\mathbf{w}|\mathscr{D})$ for weight vector $\mathbf{w}$ on the base of an observable set of data $\mathscr{D}$. We formulate

Bayes' theorem in the following form:

$$p(\mathbf{w}|\mathscr{D}) = \frac{p(\mathscr{D}|\mathbf{w})}{p(\mathscr{D})} p(\mathbf{w}). \tag{103}$$

Probability distribution $p(\mathbf{w}|\mathscr{D})$ is verbalized as "probability of $\mathbf{w}$ given $\mathscr{D}$". This means that a prior probability distribution was roughly evaluated as $p(\mathbf{w})$ and after a set of data was observed a posterior pd was estimated by means of (103). It is worth mentioning that pds are values of random variables fulfilling the normalization condition (91). That is why the evidence (93) plays the role of normalization factor to have the integral value over the variables equal unity.

The Bayesian inference predicts higher values of the posterior due to multiplication of the prior by the ratio of likelihood and evidence, see (103). This is sketched in Figure 18, where $\mathbf{w}_{\mathrm{MP}}$ is called the *Most Probable* weight vector.

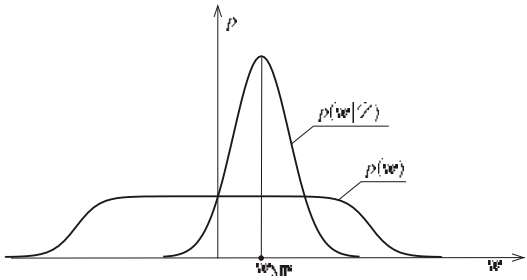


Figure 18. Estimation of probability of the most probable weight vector $\mathbf{w}_{\mathrm{MP}}$

Example 5.2 – Ockham's razor. The main goal of this example is to discuss the evaluation of model complexity.

Let us formulate now Bayes' theorem for three random variables which are: data set $\mathscr{D}$, vector of parameters $\mathbf{w}$ and a set of models $\{\mathscr{H}_l\}$. After application of sum and product rules the following form of Bayes' theorem can be derived, see e.g. MacKay (1992)

$$p(\mathbf{w}|\mathscr{D}, \mathscr{H}_l) = \frac{p(\mathscr{D}|\mathbf{w}, \mathscr{H}_l) p(\mathbf{w}|\mathscr{H}_l)}{p(\mathscr{D}|\mathscr{H}_l)}, \tag{104}$$

in which the so-called *model evidence* is written in the integral form:

$$p(\mathscr{D}|\mathscr{H}_l) = \int p(\mathscr{D}|\mathbf{w}, \mathscr{H}_l) p(\mathbf{w}|\mathscr{H}_l) d\mathbf{w}. \tag{105}$$

Let us focus on three models put in order with respect to their complexity, i.e. a simple model $\mathscr{H}_1$, a complex model $\mathscr{H}_3$ and a model $\mathscr{H}_2$ of medium complexity.

In Figure 19 the pds of these models are shown. The area under each probability distribution $p(\mathscr{D}|\mathscr{H}_l)$ equals unity but the complex model has a wider range of prediction R_3 than the simple model with range R_1. This means that the maximum value pd of complex model $p(\mathscr{D}|\mathscr{H}_3)$ is smaller than $p(\mathscr{D}|\mathscr{H}_1)$ of a simple model.

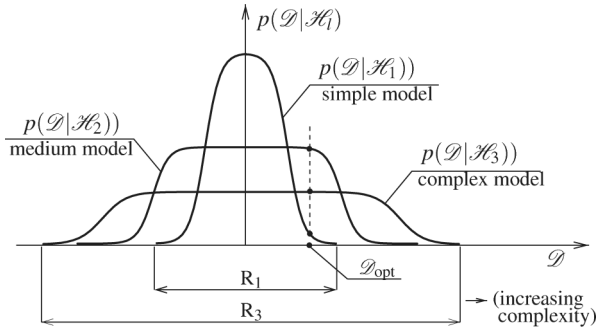


Figure 19. Conditional probability distributions of data given models $\mathscr{H}_l$

From the viewpoint of complexity the medium model $\mathscr{H}_2$ is the best, which gives satisfactory estimation of the pd value and the range of predicted data. This question will be discussed as an important problem of model design, referred also to the formulation of neural network architectures.

The application of Bayes' theorem to the evaluation of model complexity is related to the so-called *Ockham's razor* which reflects William of Ockham's sentence, expressed in the 14th century in a very theological form "non sunt multiplicanda entia sine necessitate" (from Latin "entities should not be multiplied unnecessarily"), see http://www.britannica.com/.

Now let us formulate Bayes' theorem for a simple problem of linear regression analysis without application of a penalty function for controlling the over-fitting phenomena (zero value of the hyperparameter α). Only one model is assumed to be applied so the symbol $\mathscr{H}_l$ can be omitted in (105) but the second variable, corresponding to the precision hyperparameter, so $\beta = 1/\sigma^2$ is taken into account.

Bayes' theorem can be written in the following form, where the data input set $\mathbf{X}$ and the target set $\mathbf{t}$ are explicitly written

$$p(\mathbf{w}|\mathbf{X},\mathbf{t},\beta) = \frac{p(\mathbf{t}|\mathbf{X},\mathbf{w},\beta)p(\mathbf{w})}{p(\mathbf{t}|\beta)}, \tag{106}$$

$$p(\mathbf{t}|\beta) = \int_{\mathscr{R}^W} p(\mathbf{t}|\mathbf{X},\mathbf{w},\beta)p(\mathbf{w})d\mathbf{w}. \tag{107}$$

For convenience of notation certain variables are commonly omitted, which should be formally applied in notation of conditional probabilities or in operators

applied in Bayes' theorem. The reasons for omitting the quantities can be different and except those commonly used habits, a reason of omission will be mentioned in the paper content.

A *shortened notation* gives the following form of formulae (106) and (107)

$$p(\mathbf{w}|\mathbf{t},\beta) = \frac{p(\mathbf{t}|\mathbf{w},\beta)p(\mathbf{w})}{p(\mathbf{t}|\beta)}, \tag{108}$$

$$p(\mathbf{t}|\beta) = \int p(\mathbf{t}|\mathbf{w},\beta)p(\mathbf{w})d\mathbf{w}. \tag{109}$$

In formulae (108) and (109) the input data set $\mathbf{X}$ is omitted since in the regression problems we never seek to model the given input data, and $\mathbf{X}$ is purely a conditional variable. The region of integration $\mathscr{R}^W$ is also omitted since it is clear that an appriopriate region should correspond to the variable $\mathbf{w}$.

5.4 Maximum Likelihood method

Let us turn our attention to the likelihood in the nominator of (108). The maximal value of the likelihood function $L = p(\mathbf{t}|\mathbf{w},\beta)$ gives fairly good estimation of probability of the vector of parameters $\mathbf{w}$. This approach can be deduced from the fixed value of the evidence and a lower value of the prior than the value of the posterior.

Let us assume the Gaussian (i.e. normal) probability density for all the variables except the hyperparameter $\beta = 1/\sigma^2$, which is fixed and known in advance. In later points the inference of σ^2 from the data set will be discussed. The Gaussian pd of the likelihood can be written in the following form:

$$p(\mathbf{t}|\mathbf{w},\sigma^2) = \prod_{n=1}^{N} \mathscr{N}(t^n|y(\mathbf{x}^n;\mathbf{w}),\sigma^2), \tag{110}$$

where:

$$\mathscr{N}(t^n|y(\mathbf{x}^n;\mathbf{w}),\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}[t^n - y(\mathbf{x}^n;\mathbf{w})]^2\right\}. \tag{111}$$

It was proved that it is convenient to consider the log likelihood function, see Bishop (1995, 2006):

$$\ln L \equiv \ln p(\mathbf{t}|\mathbf{w},\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}\{t^n - y(\mathbf{x}^n;\mathbf{w})\}^2 + \frac{N}{2}\ln(\frac{1}{\sigma^2}) - \frac{N}{2}\ln(2\pi). \tag{112}$$

Now we consider maximizing of $\ln L$ with respect to the model parameters, i.e. the vector $\mathbf{w}$ and variance σ^2. From the equations $\partial L/\partial \mathbf{w} = 0$ and $\partial L/\partial \sigma^2 = 0$

we obtain the following maximizers:

$$\max_{\mathbf{w}} \ln L \xrightarrow{\partial L/\partial \mathbf{w}=0} \mathbf{w}_{\mathrm{ML}} \tag{113}$$

$$\max_{\sigma^2} \Big(-\frac{1}{2\sigma^2} \sum_{n=1}^{N} \{t^n - y(\mathbf{x}^n;\mathbf{w})\}^2 + \frac{N}{2} \ln\Big(\frac{1}{\sigma^2}\Big) \Big) \xrightarrow{\partial L/\partial \sigma^2=0}$$

$$\rightarrow \sigma^2_{\mathrm{ML}} \equiv \frac{1}{N} \sum_{n=1}^{N} \{t^n - y(\mathbf{x}^n;\mathbf{w}_{\mathrm{ML}})\}^2, \tag{114}$$

where the acronym ML stands for the *Maximum of Likelihood*.

General solution for ML. The ML method corresponds to the Least Square (LS) method discussed in Point 2.2. The parameter vector $\mathbf{w}_{\mathrm{ML}}$ is chosen by mininizing the cost function $E_{\mathscr{D}}(\mathbf{w})$, defined by formula (5). This formula can be obtained as the negative log likelihood $-\ln L$ if we drop in (112) the terms independent of $\mathbf{w}$. Thus, we come to the minimization problem whose minimizer $\mathbf{w}_{\mathrm{LS}}$ can be computed in an iterative way (networks FLNN) or by the Least Square method (network RBNN)

$$\min_{\mathbf{w}} E_{\mathscr{D}}(\mathbf{w}) \equiv \max_{\mathbf{w}} \ln L =$$

$$= \min_{\mathbf{w}} \Big(\frac{1}{2} \sum_{n=1}^{N} \{t^n - y(\mathbf{x}^n;\mathbf{w})\}^2 \Big) \xrightarrow{\partial E_{\mathscr{D}}/\partial \mathbf{w}=0} \mathbf{w}_{\mathrm{LS}} \equiv \mathbf{w}_{\mathrm{ML}}. \tag{115}$$

It is worth adding that the analogue was obtained under the assumption of a Gaussian noise in the Bayesian likelihood, which is reflected in formula (110). Another interesting conclusion is that the variance of the noise σ^2_{LS}, computed from the equation $\partial E_{\mathscr{D}}/\partial \sigma^2 = 0$, equals the Mean-Square-Error (MSE), see Appendix A1:

$$MSE = \sigma^2_{\mathrm{LS}}. \tag{116}$$

The error *MSE* is explored in the training of deterministic SNNs. The only remark is that *MSE* should be computed for non-scaled output variables.

The conclusions expressed above are general and fit well the Feed-forward Layered NN (FLNN). The Radial Basis Function NN (RBFN), discussed in Point 2.4, is linear in weights so it can give explicit analytical formulae for $\mathbf{w}_{\mathrm{ML}}$ and σ^2_{ML}.

Application of RBFN in ML. Due to RBFs, the regression function $y(\mathbf{x};\mathbf{w})$ can be written in an explicit form, cf. (16):

$$y(\mathbf{x};\mathbf{w}) = \sum_{k=0}^{K} w_k \phi_k(\mathbf{x}) \equiv \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}), \tag{117}$$

where: $\mathbf{w}$ – vector of weights of dimension $(1 \times W)$ for $W = K + 1$, $\phi(\mathbf{x}) = \{\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})\}$ – vectors of RBFs of dimension $(K+1) \times 1$, where the number of RBF equals K and $\phi_0(\mathbf{x}) = 1$.

After substitution of (117) to (110) the likelihood pd takes the form:

$$L \equiv p(\mathbf{t}|\mathbf{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(t^n|\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}), \sigma^2). \tag{118}$$

From the log likelihood (112) we can deduce the error function

$$E_{\mathscr{D}}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t^n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}^n)\}^2. \tag{119}$$

The optimal weight vector $\mathbf{w}_{\mathrm{ML}}$ can be computed as the minimizer (115), which corresponds to the solution (27)

$$\mathbf{w}_{\mathrm{ML}} \equiv \mathbf{w}_{\mathrm{LS}} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}, \tag{120}$$

where $\mathbf{\Phi}$ is design matrix (24).

The mean μ_{ML} and variance σ^2_{ML} of computed and biased output variables are:

$$\mu_{\mathrm{ML}} = \mathbf{w}^{\mathrm{T}}_{\mathrm{ML}}\phi(\mathbf{x}^n), \tag{121}$$

$$\sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{t^n - \mathbf{w}^{\mathrm{T}}_{\mathrm{ML}}\phi(\mathbf{x}^n)\}^2. \tag{122}$$

5.5 Bayesian inference and MAP

General relations. Let us extend Bayes' theorem (108) with respect to adding α as an additional parameter:

$$p(\mathbf{w}|\mathbf{t}, \alpha, \beta) = \frac{p(\mathbf{t}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)}{p(\mathbf{t}|\alpha, \beta)}. \tag{123}$$

We suppose that the prior pd is Gaussian in form:

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}), \tag{124}$$

and the likelihood is given by Gaussian pd (110). The resulting posterior pd can be written in the form:

$$p(\mathbf{w}|\mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha), \tag{125}$$

which is non-Gaussian because of nonlinear dependence of $y(\mathbf{x}; \mathbf{w})$ on $\mathbf{w}$.

Substitution of Gaussian pds (110) and (124) into (125) gives an approximation of the posterior pd:

$$p(\mathbf{t}|\mathbf{w},\beta)p(\mathbf{w}|\alpha) = \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{W/2}\int \exp\{-F(\mathbf{w})\}\mathrm{d}\mathbf{w}. \tag{126}$$

The log of posterior (125) can be written in a compact form:

$$\ln p(\mathbf{w}|\mathbf{t},\alpha,\beta) = -F(\mathbf{w}) + \frac{N}{2}\ln\beta + \frac{W}{2}\ln\alpha - \frac{N+W}{2}\ln(2\pi). \tag{127}$$

Thus, omitting constant terms, the negative logarithm $-\ln p(\mathbf{w}|\mathbf{t},\alpha,\beta)$ gives the penalized cost function

$$F(\mathbf{w}) = \beta E_{\mathscr{D}}(\mathbf{w}) + \alpha E_W(\mathbf{w}) \equiv \frac{\beta}{2}\sum_{n=1}^{N}\{t^n - y(\mathbf{x}^n;\mathbf{w})\}^2 + \frac{\alpha}{2}\sum_{i=1}^{W} w_i^2. \tag{128}$$

Function (128) fully corresponds to function (8) if $\beta = 1$ and $\lambda = \alpha\sigma^2$.

Similarly as in (115), we can write the following relation:

$$\min_{\mathbf{w}} F(\mathbf{w}) = \max_{\mathbf{w}} \ln p(\mathbf{w}|\mathbf{t},\alpha,\beta) \xrightarrow{\partial F/\partial \mathbf{w}=0} \mathbf{w}_{\mathrm{MAP}}, \tag{129}$$

where the acronym MAP stands for *Maximum APosterior*.

In case of RBFN the application of minimization (129) leads to the analytical formula corresponding to (28):

$$\mathbf{w}_{\mathrm{PLS}} = [(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}) + \lambda\mathbf{I}]^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}, \tag{130}$$

where $\lambda = \alpha/\beta = \alpha\sigma^2$.

The maximization of the approximated log of posterior (127) with respect to the mean of observable values of the set of outputs $\{y(\mathbf{x}^n,\mathbf{w}_{\mathrm{MAP}})\}$ and with respect to the variance $\sigma^2 = 1/\beta$ gives the following formulae:

$$\mu_{\mathrm{MAP}} = \frac{1}{N}\sum_{n=1}^{N} y(\mathbf{x}^n,\mathbf{w}_{\mathrm{MAP}}), \tag{131}$$

$$\sigma^2_{\mathrm{MAP}} = \frac{1}{N}\sum_{n=1}^{N}\{t^n - \mathbf{w}^{\mathrm{T}}_{\mathrm{MAP}}\phi(\mathbf{x}^n)\}^2. \tag{132}$$

The mean μ_{MAP} and variance σ^2_{MAP} can be related to observable output points (biased data). In case we remove bias from observable data the factor $N-1$ is taken into account in (132) and unbiased variance takes the form, cf. Bishop (2006), pp. 27 and 171:

$$\tilde{\sigma}^2_{\mathrm{MAP}} = \frac{1}{N-1}\sum_{n=1}^{N}\{t^n - \mathbf{w}^{\mathrm{T}}_{\mathrm{MAP}}\phi(\mathbf{x}^n)\}^2. \tag{133}$$

Other formulae can be also obtained from maximization of (127) with respect to hyperparameters α and β:

$$\alpha_{\text{eff}} = \frac{W}{2E_{\text{W}}(\mathbf{w}_{\text{MAP}})}, \quad \beta_{\text{eff}} = \frac{N}{2E_{\mathscr{D}}(\mathbf{w}_{\text{MAP}})}. \tag{134}$$

Parameters (134) are called *efficient hyperparameters*. They can be used in an iterative procedure for improving the weight vector $\mathbf{w}_{\text{MAP}}$ computed by standard (deterministic) NNs, cf. Bishop (1995), Chapter 10.

Linear predictive distribution and curve fitting. Let us assume that the values of hyper-parameters α and β are fixed. Results obtained above were obtained on the base of point estimation of the vector $\mathbf{w}_{\text{MAP}}$ in regression. From a practical point of view we are interested in the prediction of a regression curve in points other than those corresponding to training patterns, i.e. in fact, a continuous regression curve $y(\mathbf{x};\mathbf{w}_{\text{MAP}})$ is needed. This question is related to a more Bayesian approach in which the conditional probability distribution (100) is given in form of the integral *over all weights* w_i, written in full notation:

$$p(t^*|\mathbf{x}^*,\mathbf{X},\mathbf{t},\alpha,\beta) = \int_{\mathscr{R}^W} p(t^*|\mathbf{x}^*,\mathbf{X},\mathbf{w},\beta)p(\mathbf{w}|\mathbf{X},\mathbf{t},\alpha,\beta)d\mathbf{w}, \tag{135}$$

where: $(\mathbf{x}^*,t^*)$ – continuous variables, called a single, "new" or prediction pattern, $\mathbf{X} = \{\mathbf{x}^1,\ldots,\mathbf{x}^N\}, \mathbf{t} = \{t^1,\ldots,t^N\}$ – sets of training data.

In the Bayesian analysis we apply the so-called *principle of marginalization*, i.e. integration over marginal variables (variables we want to eliminate). In the given conditional probability distribution (135) the weight vector $\mathbf{w}$ is treated as a marginal variable.

In case of RBFN model the conditional distribution (135) is taken for a single pattern $(\mathbf{x},t)$. Then, taking into account Bayesian laws, the following *predictive distribution* for predicting a new pattern $(\mathbf{x}^*,t^*)$ can be obtained, see Bishop (1995), p. 400:

$$p(t^*|\mathbf{x}^*,\mathbf{X},\mathbf{t},\alpha,\beta) = \mathscr{N}\{t|\mathbf{w}_{\text{MAP}}^{\text{T}}\boldsymbol{\phi}(\mathbf{x}^*),\sigma_N^2(\mathbf{x}^*)\}, \tag{136}$$

The variance $\sigma_N^2(\mathbf{x}^*)$ for predictive distribution is given by the curves corresponding to standard deviation $\sigma_N(\mathbf{x}^*)$, in literature called *error sigma bar* σ_{t}, see Bishop (2006), p. 156:

$$\sigma_N^2(\mathbf{x}^*) \equiv \sigma_{\text{t}}^2(\mathbf{x}^*) = \sigma^2 + \boldsymbol{\phi}(\mathbf{x}^*)^{\text{T}}\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x}^*). \tag{137}$$

where:

$$\mathbf{S}_N = (\beta\boldsymbol{\Phi}^{\text{T}}\boldsymbol{\Phi} + \alpha\mathbf{I})^{-1}. \tag{138}$$

The first term in (137) corresponds to the noise of data. The second term represents the uncertainty related to the vector of parameters $\mathbf{w}$. Because the pds in $\mathbf{w}$

are independent it was possible to prove (see references in Bishop (2006), p.156) that $\sigma^2_{N+1}(\mathbf{x}) \leq \sigma^2_N(\mathbf{x})$ and for $N \to \infty$ the $\sigma^2_N(\mathbf{x}) \to 1/\beta = \sigma^2$.

Example 5.3, taken from Bishop (2006), pp.157-158. For illustration the application of predictive distribution for the Bayesian linear regression model (basing on the interpolation RBFN), we return to the data set generated from the sinusoidal mathematical curve $h(x) = \sin 2\pi x$.

In Figure 20 four special cases are shown, corresponding to N = 1, 2, 4 and 10 patterns. The broken line represents the mathematical regression curve $h(x)$.

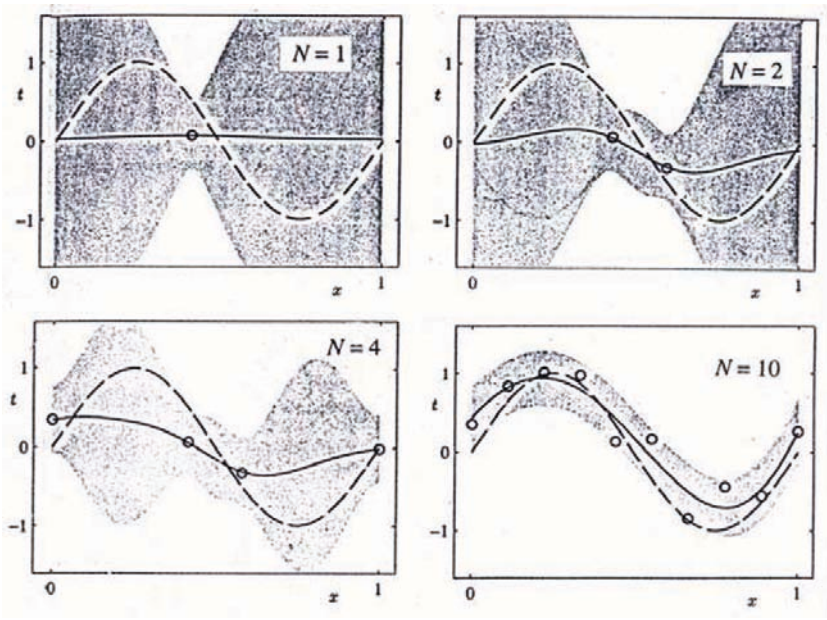


Figure 20. Fitting curves and regions of $\pm\sigma_N$ uncertainty computed either by the Gaussian model consisting K = 9 Gaussian RBFs (for N = 1, 2, 4 patterns) and polynomial RBF for N = 10

The Bayesian prediction was carried out by means of a model consisting of $K = 9$ Gaussian RBFs. This model was used for three data sets composed of $N = 1, 2, 4$ patterns. The fourth figure was made by the polynomial model of order $K = 9$. The computations of fitting curves $y(x; \mathbf{w}_{\mathrm{MAP}})$, plotted as continuous lines in Figure 20, were carried out for fixed values of hyperparameters $\alpha = 5 \times 10^{-3}$ and $\beta = 1/\sigma^2 = 11.1$. The shaded region of the prediction uncertainty spans on 1 standard deviation bounds (1 sigma error bars $1\sigma_{\mathrm{t}}(x) = \sigma_N(x)$ around the mean $y(x; \mathbf{w}_{\mathrm{MAP}})$).

It is worth noting that the predictive region of uncertainty depends on x, ac-

cording to (137). The width of this region equals $2\sigma_N(x)$ and, in case of Gaussian RBFs, is smallest in the neighbourhood of the data points where centres of RBFs were placed. It is in these points of independent variables $x^n = \mu_k$ that the bounds are nearly $\pm 1\sigma$. The same situation occurs if the number of patterns increases. This is the case of $N = 10$ patterns, analysed by $K = 9$ polynomial RBFs (7). It is visible that the fitting curve (polynomial of the 9th order) is bounded by nearly equidistant curves: $\pm 1\sigma_N \approx \pm(1\sigma_t = 0.3)$.

5.6 A more general Bayesian framework

Computation of predictive distribution. Let us return to the problem of predicting a single variable t^*, corresponding to an input $\mathbf{x}^*$. The conditional Gaussian distribution can be written in a general form

$$p(t^*|\mathbf{x}^*, \mathbf{w}, \beta) = \mathcal{N}(t^*|y(\mathbf{x}^*; \mathbf{w}), \beta^{-1}). \tag{139}$$

This formula is referred to a general form of regression function $y(\mathbf{x}; \mathbf{w})$ which can be used as an output of the Feed-forward Layered Neural Network (10). Now the regression problem is nonlinear, contrary to the network with RBFs which imply the "linearity in weights".

Using distribution $p(\mathbf{w}|\alpha)$ in form (124) we can obtain the posterior $p(\mathbf{w}|\mathscr{D}, \alpha, \beta)$ in the form corresponding to (125) and the log posterior takes the form (127). Because the function $y(x; \mathbf{w})$ is nonlinear, the solution $\mathbf{w}_{\text{MAP}}$ can be found by means of neural networks, applying error backpropagation or any other learning method.

Having evaluated $\mathbf{w}_{\text{MAP}}$ we can use it in the Gaussian approximation to the posterior distribution, see Bishop (2006), p.279:

$$q(\mathbf{w}|\mathscr{D}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{A}^{-1}), \tag{140}$$

where matrix $\mathbf{A}$ is given by the formula

$$\mathbf{A} = -\nabla_{\mathbf{w}} \nabla_{\mathbf{w}} p(\mathbf{w}|\mathscr{D}, \alpha, \beta) = \beta \mathbf{H} + \alpha \mathbf{I}. \tag{141}$$

In formula (141) the *Hessian matrix* $\mathbf{H}$ appears comprising the second derivatives of the error function $E_{\mathscr{D}}$ with respect to the components of weight vector $\mathbf{w}$:

$$\mathbf{H} = \nabla_{\mathbf{w}} \nabla_{\mathbf{w}} E_{\mathscr{D}}. \tag{142}$$

Predictive distribution is obtained by application of the marginalization principle:

$$p(t^*|\mathbf{x}^*, \mathscr{D}, \alpha, \beta) = \int p(t^*|\mathbf{x}^*, \mathbf{w}) q(\mathbf{w}|\mathscr{D}, \alpha, \beta) d\mathbf{w}. \tag{143}$$

However, even with the Gaussian approximation to the posterior, the *integration is analytically intractable* because of the neural network function $y(\mathbf{x}; \mathbf{w})$.

Thus, we have to either apply the numerical integration or use approximation of $y(\mathbf{x};\mathbf{w})$.

The common approach is to use the Taylor series and restriction to the linear approximation of $y(\mathbf{x};\mathbf{w})$, which leads to the formula

$$y(\mathbf{x};\mathbf{w}) \cong y(\mathbf{x};\mathbf{w}_{\mathrm{MAP}}) + \mathbf{g}^{\mathrm{T}}(\mathbf{w}-\mathbf{w}_{\mathrm{MAP}}), \tag{144}$$

where

$$\mathbf{g} = \nabla_{\mathbf{w}} y(\mathbf{x};\mathbf{w})|_{\mathbf{w}=\mathbf{w}_{\mathrm{MAP}}}. \tag{145}$$

The approximation (145) gives the following Gaussian pd of predictive distribution, see Bishop (2006), p.279:

$$p(t^*|\mathbf{x}^*,\mathscr{D},\alpha,\beta) = \mathscr{N}(t^*|y(\mathbf{x}^*;\mathbf{w}_{\mathrm{MAP}}),\sigma_{\mathrm{t}}^2(\mathbf{x}^*)), \tag{146}$$

where the input-dependent variance is given by

$$\sigma_{\mathrm{t}}^2 = \sigma^2 + \mathbf{g}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{g}, \tag{147}$$

which is commonly used for computation of the error sigma bar σ_{t}.

In case of application of RBFN, the analysis becomes linear and corresponding analytical formulae (136) and (137) can be applied.

The main problem of the general Bayesian inference is the application of the Hessian matrix (142). The computation of the second order error function gradients needs more operations so very often simple approximations are used in the neural network learning methods. From among many approximations, let us mention only the use of the first gradients of the error function in the pseudo-Newtonian learning methods applied in ANNs.

Let us write the Hessian matrix in the form

$$\mathbf{H} = \nabla\nabla E_{\mathscr{D}} \equiv \sum_{n=1}^{N} \nabla y^n \nabla y^n + \sum_{n=1}^{N} (y^n - t^n)\nabla\nabla y^n. \tag{148}$$

where $\nabla y^n = \partial(y(\mathbf{x}^n);\mathbf{w})/\partial\mathbf{w}$. The elements of the matrix $\mathbf{H}$ can be found in $O(W^2)$ steps by simple multiplication. This estimation can be diminished to $O(W)$ steps applying the *approximation of the Hessian matrix*

$$\mathbf{H} \cong \sum_{n} \mathbf{b}_n \mathbf{b}_n^{\mathrm{T}}, \tag{149}$$

where $\mathbf{b}_n = \nabla y^n$. This approach is used in the Levenberg-Marquardt learning method but for general network mapping the second term in (148) cannot be omitted.

Marginal likelihood. Let us return to Bayes' theorem (123) writing the evidence

$p(\mathbf{t}|\alpha,\beta)$ in an extended form $p(\mathscr{D}|\alpha,\beta)$. Then we express it by means of the sum rule in the following integral form

$$p(\mathscr{D}|\alpha,\beta) = \int p(\mathscr{D}|\mathbf{w},\beta)p(\mathbf{w}|\alpha)d\mathbf{w}, \tag{150}$$

which is the *marginal likelihood*. Formula (150) is obtained from the likelihood via marginalization of the weight parameter vector $\mathbf{w}$. After substitution Gaussian pd (111), the formula (150) takes the form

$$p(\mathscr{D}|\alpha,\beta) = (\beta/2\pi)^{N/2}(\alpha/2\pi)^{W/2}\int \exp(-F(\mathbf{w}))d\mathbf{w}, \tag{151}$$

where the function $F(\mathbf{w})$ is the error cost function

$$F(\mathbf{w}) = F(\mathbf{w}_{\mathrm{MAP}}) + \frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MAP}})^{\mathrm{T}}\mathbf{A}(\mathbf{w}-\mathbf{w}_{\mathrm{MAP}}). \tag{152}$$

The function $F(\mathbf{w}_{\mathrm{MAP}})$ corresponds to formula (128):

$$\begin{aligned} F(\mathbf{w}_{\mathrm{MAP}}) &= \beta E_{\mathscr{D}}(\mathbf{w}_{\mathrm{MAP}}) + \alpha E_W(\mathbf{w}_{\mathrm{MAP}}) = \\ &= \frac{\beta}{2}\sum_{n=1}^{N}\{t^n - y(\mathbf{x}^n;\mathbf{w}_{\mathrm{MAP}})\}^2 + \frac{\alpha}{2}\mathbf{w}_{\mathrm{MAP}}^{\mathrm{T}}\mathbf{w}_{\mathrm{MAP}}. \end{aligned} \tag{153}$$

In the following analysis we will apply the approximate *log marginal likelihood*

$$\ln p(\mathscr{D}|\alpha,\beta) \cong -F(\mathbf{w}_{\mathrm{MAP}}) - \frac{1}{2}\ln|\mathbf{A}| + \frac{W}{2}\ln\alpha + \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi. \tag{154}$$

In case of application of Gaussian RBFs the $F(\mathbf{w})$ function takes the following analytical form:

$$\begin{aligned} F(\mathbf{w}) &= \frac{\beta}{2}||\mathbf{t}-\mathbf{\Phi}\mathbf{w}||^2 + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} = \\ &= F(\mathbf{m}_N) + \frac{1}{2}(\mathbf{w}-\mathbf{m}_N)^{\mathrm{T}}\mathbf{S}_N^{-1}(\mathbf{w}-\mathbf{m}_N), \end{aligned} \tag{155}$$

where:

$$F(\mathbf{m}_N) = \frac{\beta}{2}||\mathbf{t}-\mathbf{\Phi}\mathbf{m}_N||^2 + \frac{\alpha}{2}\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N, \tag{156}$$

$$\mathbf{S}_N^{-1} = \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \alpha\mathbf{I}, \tag{157}$$

$$\mathbf{m}_N = \beta\mathbf{S}_N\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}. \tag{158}$$

In Figure 21 the plot of the negative log marginal likelihood $-\ln Mal(\ln\alpha)$ is shown (the term const(N,β)) is omitted). The most striking effect is that the *function* $-\ln Mal(\ln\alpha)$ *has a minimum*! It was evaluated that the minimal value of

$$\min_{\mathbf{w}}(-\ln Mal(\ln\alpha)) \rightarrow \ln\alpha_{\text{Mal}} = 0.8. \tag{160}$$

A great value of the conclusion expressed above is that the log marginal likelihood curve $\ln Mal(\ln\alpha)$ has the maximum at $\ln\alpha_{\text{Mal}}$ (minimum for $-\ln Mal(\ln\alpha)$). Thus, the curve of log marginal likelihood can be applied for the selection of an optimal value of the regularization parameter $\ln\alpha_{\text{Mal}}$. The *criterion MML* (Maximum Marginal Likelihood) *can play a crutial role* since on the same set of training patterns it is possible to optimize the numerical model only with respect to the training patterns (*without a validation set of patterns*!). This conclusion can be invaluable especially for small known set of patterns $\mathscr{P}$ (in such a case the training set can correspond to the full set $\mathscr{L} = \mathscr{P}$).

Example 5.6, taken from Bishop (2006), pp. 167-168. Let us now apply the criterion of maximum marginal likelihood to the evaluation of an optimal order K of the polynomial. The number of training patterns is $N = 10$ and values of hyperparameters are fixed as $\beta = 1, \alpha = \lambda = 5\times 10^{-3}$.

The log marginal likelihood (159) can be used and its value is computed assuming $K = 0, 1, \ldots, 9$. In Figure 22 the plot of $\ln Mal(K;\mathscr{L})$ is shown versus the error function curves $RMS(K;\mathscr{S})$, where $\mathscr{S} = \mathscr{L}$ corresponds to the training set of patterns and $\mathscr{S} = \mathscr{V}$ to a validation set, cf. Bishop (2006), Figure 1.5.

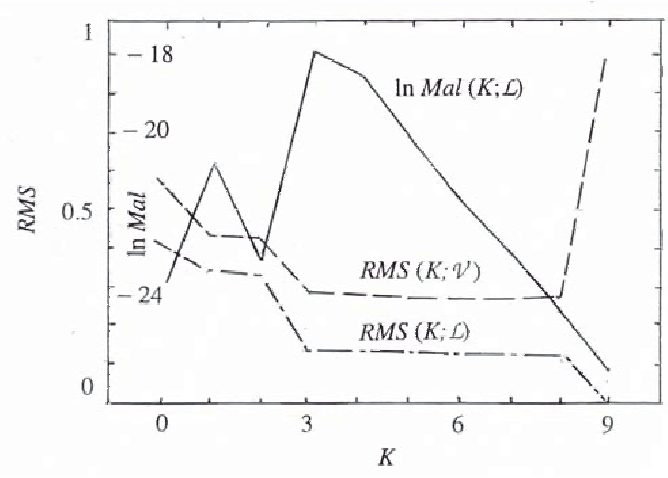


Figure 22. Plots of error functions $RMS(K;\mathscr{S})$ for training $(\mathscr{S} = \mathscr{L})$ and validation set $(\mathscr{S} = \mathscr{V})$, vs. log marginal likelihood curve $\ln Mal(K;\mathscr{L})$

The graphs plotted for the $K = 0,1,2$ polynomials (7) give a poor approximation if we look at the training and validation curves $RMS(K;\mathscr{L})$ and $RMS(K;\mathscr{V})$, respectively. Then for the polynomial orders between $K = 3$ and $K = 8$ the errors are roughly constant. The log marginal likelihood curve $\ln Mal(K;\mathscr{L})$, computed for the training set of patterns shows an increase of ln Mal values changing the polynomial order from K = 0 to K = 1. The polynomial gives a decrease of $\ln Mal$ value for $K = 2$ since the quadratic polynomial has no even terms for good approximation of the sinusoidal curve. The global maximum is reached for $K = 3$. The criterion of maximum of the log marginal function clearly indicates the cubic polynomial as a simple model, contrary to the cross-validation criterion which gives nearly the same values of the validation error for $K \in [3,8]$.

The remarks expressed above are of great practical value. The maximum of $\ln Mal$ prefers a model which is neither too simple nor too complex. This conclusion fully agrees with Ockham's razor criterion. Moreover, the maximum log *Mal* can indicate the best model, contrary to the cross-validation criterion which can give preferences to some models of various complexity.

The marginal likelihood (150) fully corresponds to the model evidence in (123). The searching for Maximum of Marginal Likelihood is performed by means of the *Evidence Procedure*, see Nabney (2004). In this procedure hyperparameters are updated using their optimal values.

5.7 SNN/MAP networks

Optimization of hyperparameters. So far, hyperparameters α and β have been assumed as fixed and known in advance. The nonlinear Bayesian analysis is based on hyperparameters which depend on the data set, values of noises or other approximation errors. Thus, in the full nonlinear analysis hyperparameters are now variable. Corresponding optimal values of α_{opt} and β_{opt} can be derived on the base of a more general Bayesian framework.

In the Evidence Procedure we can compute point estimations of α and β by maximizing $\ln p(\mathscr{D}|\alpha,\beta)$. We will refer the analysis to the eigenvalues of the Hessian matrix. Let us start with the following eigenequation

$$\beta \mathbf{H}\mathbf{u}_i = \lambda_i \mathbf{u}, \tag{161}$$

where $i = 1,\ldots,W$ for W corresponding to the weight parameter space dimension. It can be proved, see Bishop (2006) pp.169 and 280-281, that maximization of (154) with respect to α and β gives the following two formulae:

$$\alpha_{\text{opt}} = \frac{\gamma}{\mathbf{w}_{\text{MAP}}^{\text{T}}\mathbf{w}_{\text{MAP}}}, \tag{162}$$

$$\frac{1}{\beta_{\text{opt}}} \equiv \sigma_N^2 = \frac{1}{1-\gamma}\sum_{n=1}^{N}\{t^n - y(\mathbf{x}^n;\mathbf{w}_{\text{MAP}})\}^2, \tag{163}$$

where

$$\gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\alpha + \lambda_i}. \tag{164}$$

Formulae (162) to (164) reflect the coupling of the hyperparameters α_{opt} and β_{opt} and the computed (updated) weight vector $\mathbf{w}_{\text{MAP}}$ by means of the parameter γ. In case $\lambda_i << \alpha$ we have $\gamma \to 0$ and the precision parameter $\beta_{\text{MAP}} \approx \beta_{\text{ML}}$. The other case $\lambda_i >> \alpha$ gives $\gamma \to W$.

The case of $N >> W$ corresponding to a great number of observations is worth mentioning. If additionally the parameters are well determined, i.e. for $\lambda_i >> \alpha$ then formulae (134) are valid since $\alpha_{\text{opt}} \to \alpha_{\text{eff}}$ and $\beta_{\text{opt}} \to \beta_{\text{eff}}$.

SNN/MAP networks. In practical implementation, we can apply the Evidence Procedure in the SNN training process. Hyperparameters are used in the penalized error function $F(\mathbf{w})$ given by formula (128) with hyperparameters α and β.

We can start from certain initial values of α_{in} and β_{in} and applying a learning method SSN we can compute the weight vector $\mathbf{w}_{\text{MAP}}^{\text{old}}$. Then the hyperparameters are updated by means of formulae:

$$\alpha_{\text{new}} = \frac{\gamma}{2E_{\text{W}}(\mathbf{w}_{\text{MAP}}^{\text{old}})}, \quad \beta_{\text{new}} = \frac{N - \gamma}{2E_{\mathscr{D}}(\mathbf{w}_{\text{MAP}}^{\text{old}})}. \tag{165}$$

Then the values of hyperparameters (165) are introduced to the supervised learning of SSN for computing the parameter $\mathbf{w}_{\text{MAP}}^{\text{new}}$. To continue iteration we substitute $\mathbf{w}_{\text{MAP}}^{\text{new}} \to \mathbf{w}_{\text{MAP}}^{\text{old}}$ into the recursive formula (165).

Instead of (165) the approximate formulae (134) can be also applied. This enables us to omit the eigenanalysis of the Hessian matrix (161).

The approach discussed above corresponds to formulation of a new learning method of standard neural networks. The Bayesian framework is explored for optimization of hyperparameters used in the penalized cost function $F(\mathbf{w})$. Thus, the merit of the neural network is not change since the minimization of the cost function error is the paradigm of the network SSN/MAP.

The SSN/MAP networks are discussed in the book by Bishop (1995). The application of these networks to the analysis of various problems, cf. e.g. Foresee and Hagan (1997), Słoński (2005) indicates out that only several updates of hyperparameters are needed to obtain satisfactory results. SNN/MAPs were also applied to the analysis of engineering problems discussed in Points 6.1 and 6.4.

5.8 General Bayesian analysis

Practical Bayesian prediction and a Monte Carlo method. In the basic Bayes' theorem (123) we used hyperparameters only as conditional variables. The theo-

rem can be written in a more practical form

$$p(\mathbf{w},\alpha,\beta|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w},\beta)p(\mathbf{w}|\alpha)p(\alpha)p(\beta)}{p(\mathbf{t})}, \tag{166}$$

where the evidence (marginal likelihood) is

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{w},\beta)p(\mathbf{w}|\alpha)p(\alpha)p(\beta)\mathrm{d}\mathbf{w}\mathrm{d}\alpha\mathrm{d}\beta. \tag{167}$$

Comparing (167) and (150) we can see that marginalization is extended. Now not only the weight parameter vector $\mathbf{w}$ but also hyperparameters α and β are treated as marginal variables. Therefore, now the prior pds $p(\alpha)$ and $p(\beta)$ should be defined. The pd $p(\alpha)$ is so-called *hyperprior*.

Similarly to (167) the Bayesian predictive distribution can be written in the form

$$p(t^*|\mathbf{t}) = \int p(t^*|\mathbf{w},\beta)p(\mathbf{w},\alpha,\beta|\mathbf{t})p(\alpha)p(\beta)\mathrm{d}\mathbf{w}\mathrm{d}\alpha\mathrm{d}\beta. \tag{168}$$

Unfortunately, nearly always *integrals* (167) and (168) *cannot be analytically* tractable to compute. There are *two ways* possible to the *integration over marginal variables*. The *first approach focuses on approximations* which enable obtaining the posteriors in the form of analytical probability distribution. Some ideas of this approach are discussed briefly in the next Point.

The *second approach*, commonly used, is related to the application of *numerical algorithms*. One of more efficient techniques is to join the numerical Monte Carlo method with numerical algorithms of data sampling. Below we present in short the main ideas of the *Markov Chain Monte Carlo* (MCMC) method, see Bishop (2006), pp. 537-554, whose modification is known as the Hybrid MC method, see Neal (1992).

Let us consider an integral and its Monte Carlo simulation

$$I = \int F(\mathbf{w})p(\mathbf{w}|\mathscr{D})\mathrm{d}\mathbf{w} \cong \frac{1}{MC}\sum_{i=1}^{MC} F(\mathbf{w}_i)p(\mathbf{w}_i|\mathscr{D}) = \\ = \frac{1}{MC}\sum_{i=1}^{MC}\frac{\sum_i F(\mathbf{w}_i)\tilde{p}(\mathbf{w}_i|\mathscr{D})}{\sum_i \tilde{p}(\mathbf{w}_i|\mathscr{D})} \tag{169}$$

where: $p(\mathbf{w}|\mathscr{D})$ – posterior pd of $\mathbf{w}$, MC – the number of Monte Carlo samples, $\tilde{p}(\mathbf{w}_i|\mathscr{D})$ un-normalized distribution.

The MCMC method is a *random walk* in which the successive steps are attained adding a small random noise ε to the weight parameter vector $\mathbf{w}$:

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + \varepsilon. \tag{170}$$

The walk is assumed to obey the first order Markov chain, defined in Point 4.1. In case of the *Hybrid Monte Carlo* (HMC) the gradient of $p(\mathbf{w}|\mathscr{D})$ is applied to choose search directions which favour regions of high posterior pd.

The candidate steps are controlled by the application of a sampling method. What is commonly used are either the *Metropolis-Hastings* or *Gibbs* method, see e.g. procedures in the manual of NETLAB by Nabney (2004). The main idea to control the candidate step is, see Bishop (1995), p. 427:

$$\begin{aligned} &\text{a) if } p(\mathbf{w}_{\text{new}}|\mathscr{D}) \geq p(\mathbf{w}_{\text{old}}|\mathscr{D}) \text{ then accept candidate sample,} \\ &\text{b) if } p(\mathbf{w}_{\text{new}}|\mathscr{D}) < p(\mathbf{w}_{\text{old}}|\mathscr{D}) \text{ then reject candidate sample.} \end{aligned} \tag{171}$$

In Figure 23 an example, taken from Bishop (2006), p. 539, is presented. It illustrates the application of the Metropolis-Hastings method, in which the fulfilling condition (171) led to rejection of 49 samples from among 150 generated candidate samples.

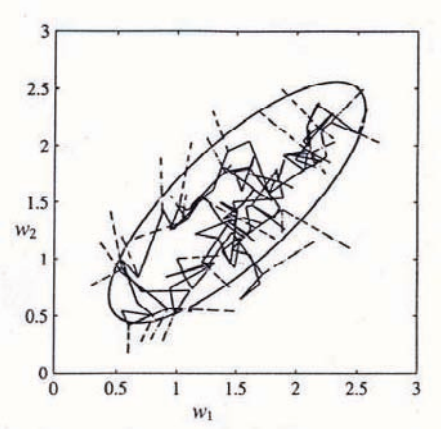


Figure 23. Illustration of the Metropolis algorithm in which rejected candidates are marked by broken lines

Type–II ML approximation. From among analytical methods suggested to be applied to marginalize certain probability densities, the maximum of Type-II Marginal Likelihood (see Tipping (2004)) is below discussed briefly.

Let us start from the product rule of probability and write down the following relation

$$p(\mathbf{w}, \alpha, \beta|\mathbf{t}) \equiv p(\mathbf{w}|\mathbf{t}, \alpha, \beta)p(\alpha, \beta|\mathbf{t}). \tag{172}$$

The first term in (172) is known as $p(\mathbf{w}|\mathbf{t}, \alpha, \beta) \sim \mathscr{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and the second term can be approximated by δ-function at its mode, i.e. we find the "*most proba-*

ble" values α_{MP} and β_{MP} which maximize the posterior, see Tipping (2004):

$$p(\alpha,\beta|\mathbf{t}) = \frac{p(\mathbf{t}|\alpha,\beta)p(\alpha)p(\beta)}{p(\mathbf{t})}. \tag{173}$$

Since the denominator is independent of α and β, we maximize the nominator of (173). Furthermore, if we assume very flat priors, called *uninformative priors* over $\ln\alpha$ and $\ln\beta$, then we come to the problem of the maximum of marginal likelihood.

Having found α_{MP} and β_{MP} we can formulate the following approximation for predictive distribution

$$p(t^*|\mathbf{t}) \cong \int p(t^*|\mathbf{w},\beta_{\mathrm{MP}})p(\mathbf{w}|\mathbf{t},\alpha_{\mathrm{MP}},\beta_{\mathrm{MP}})\mathrm{d}\mathbf{w}. \tag{174}$$

The integral (174) is computable and it is a Gaussian pd:

$$p(t^*|\mathbf{t}) = \mathcal{N}\{\mu^*,\sigma_*^2)\}. \tag{175}$$

The mean and variance of (5.8) can be analytically expressed in case of RBFs application:

$$\mu^* = \mathbf{w}_{\mathrm{MP}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}^*), \quad \sigma_*^2 = \sigma_{\mathrm{MP}}^2 + \boldsymbol{\phi}(\mathbf{x}^*)^{\mathrm{T}}\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x}^*), \tag{176}$$

where the vector $\boldsymbol{\phi}(\mathbf{x}^*)$ and the matrix $\mathbf{S}_N$ are defined in (23) and (138), respectively.

Sparse Bayesian models and hierarchical priors. A lot of attention in the linear analysis has recently been paid to the sparse learning algorithms (Bishop (2006), Tipping (2004)). These algorithms set many weight w_k to zero in the estimator function $y(\mathbf{x}) = \sum_k \phi_k(\mathbf{x})w_k$.

The sparsity can be attained, in fact, by introducing a so-called *hierarchical prior*, related to the vector of hyperparameters $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^W$. This prior and β prior are defined by means of Gamma distribution, see Bishop and Tipping (2003):

$$p(\boldsymbol{\alpha}) = \prod_{i=1}^{W}\mathrm{Gamma}(\alpha_i|a,b), \quad p(\beta) = \mathrm{Gamma}(\beta|c,d), \tag{177}$$

where

$$\mathrm{Gamma}(\alpha|a,b) = \frac{1}{\Gamma(a)}b^a\alpha^{a-1}e^{-b\alpha}, \tag{178}$$

in which $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}\mathrm{d}t$ is the Gamma function. It was proved by Bishop and Tipping (2003) that it is possible to assume zero values for the scaling parameters $a = b = c = d = 0$.

The *effective prior* $p(w_i)$ is defined by a Student's t-distribution

$$p(w_i) = \int p(w_i|\alpha_i)p(\alpha_i)\mathrm{d}\alpha_i = \frac{b^a\Gamma(a+1/2)}{(2\pi)^{1/2}\Gamma(a)}(b+w_i^2/2)^{-(a+1/2)}. \tag{179}$$

In Figure 24a the Gaussian distribution $p(\mathbf{w}|\alpha)$ in two-dimensional domain $p(w_1,w_2)$ is shown. In Figure 24b the Student's t-distribution prior $p(\mathbf{w})$ is presented after hyperparameters and the Gamma pd product were integrated. As can be seen in Figure 24b, the probability mass is concentrated close to the origin where both weights go to zero, and also along the ribs where one of the two weights goes to zero.

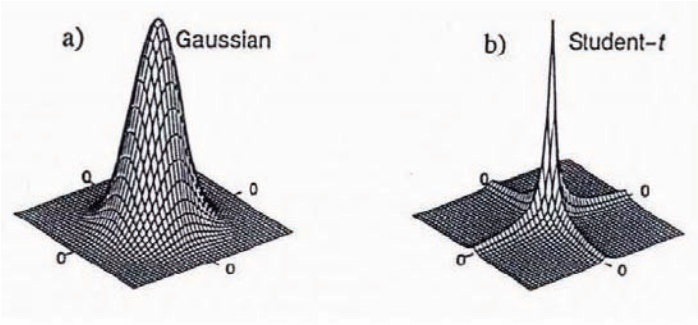


Figure 24. a) Gaussian prior $p(\mathbf{w}|\alpha)$, b) Student's t-distribution for prior $p(\mathbf{w})$.

5.9 Kernels and Gaussian Process

Kernels methods. In Point 2.4 we briefly discussed the RBFN network in which the Gaussian RBFs are used. The simplest Gaussian RBF can be written in the form in which we can use the Euclidean distance $||\mathbf{x}-\mathbf{x}'||$:

$$\phi'(\mathbf{x}) = \exp(-||\mathbf{x}-\mathbf{x}'||/2\sigma^2), \tag{180}$$

where $\mathbf{x}'$ is the centre of RBF. Instead of (180) we can formulate a so-called *kernel function*

$$k(\mathbf{x},\mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{x}') \equiv k(||\mathbf{x}-\mathbf{x}'||). \tag{181}$$

The main idea of the kernel framework is introduction of the inner product in the input space, which allows formulation of interesting extensions of many well-known algorithms. The kernels can be of different forms. For instance, the tanh sigmoid function can be written as the following sigmoidal kernel:

$$k(\mathbf{x},\mathbf{w}=\mathbf{x}') = \tanh(a\mathbf{x}^{\mathrm{T}}\mathbf{x}'+b). \tag{182}$$

Kernels can join the exponential and algebraic functions. For instance, in the GP analysis the following kernel is applied:

$$k(\mathbf{x}^n,\mathbf{x}^m)=\theta_0\exp\{\frac{\theta_1}{2}||\mathbf{x}^n-\mathbf{x}^m||^2\}+\theta_2+\theta_3(\mathbf{x}^n)^{\mathrm{T}}\mathbf{x}^m, \tag{183}$$

where: $\mathbf{x}^n,\mathbf{x}^m$ – input pattern points, $\boldsymbol{\theta}=\{\theta_i\}_{i=0}^3$ – set of the input space parameters.

The kernel $k(\mathbf{x}^n,\mathbf{x}^m)$ is a component of design matrix known as *Gramm matrix*

$$\mathbf{K}_{N\times N}=\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}=[k(\mathbf{x}^n,\mathbf{x}^m)], \tag{184}$$

where: $\boldsymbol{\Phi}^{\mathrm{T}}=[\phi(\mathbf{x}^1),\ldots,\phi(\mathbf{x}^N)]$. Please turn attention that now $\boldsymbol{\Phi}$ is a vector in (184), not the design matrix (24).

Kernel functions can be used as RBFs in all the above Points where the linear regression problems have been discussed. Kernel models are a basis for building a numerical model called the *Support Vector Machine* (SVM). In the book by Bishop (2006), pp.339-343, SVMs are in short discussed. This numerical model is similar to the interpolation of RBFN algorithm and it can be efficiently applied in the linear regression analysis. In Figure 25a, taken from Bishop (2006), p.344, an illustration is shown of the application of seven kernel functions (double circle points) for the case study of the sinusoidal synthetic data with ten points (single circles).

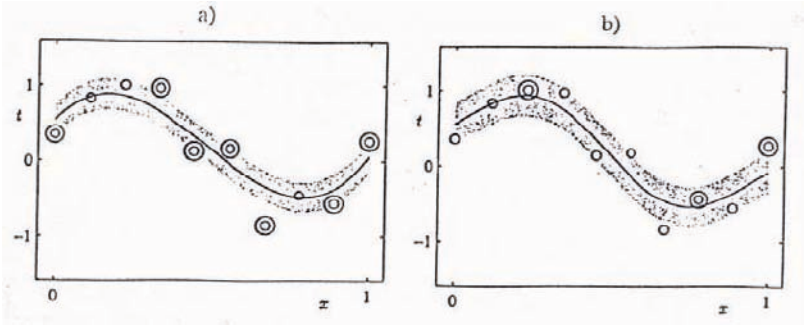


Figure 25. Predicting regression curves and 1sigma error bars for $N=10$ sinusoidal synthetic data, obtained by means of: a) Support Vector Machine (SVM), b) Relevant Vector Machine (RVM)

On the base of sparse Bayesian models Tipping (2001) formulated a modification of SVM called the *Relevance Vector Machine* (RVM). Due to computational advantages of RVM the number of corresponding kernels can be significantly decreased, cf. Figure 25b where only three kernels were applied, vs. seven kernels in SVM.

Gaussian Process in Bayesian prediction. On the base of kernel models the *Gaussian Process* (GP) approach was developed and recently it has been introduced to the Bayesian prediction, see e.g. review by MacKay (1998) and books by Bishop (2006), Rasmussen and Williams (2006). GP dispenses with the parametric model and, defines instead, directly a prior pd over functions. The main idea of GP was clearly explained in a paper by Bailer-Jones et al. (1997). That is why we can start from Figure 26, taken from the paper mentioned above (one-dimensional input and output spaces are considered).

On the base of data $\mathscr{D}$ composed of N patterns (see Figure 26 where $N = 4$ patterns) we wish to predict a new pattern point of the target value t^{N+1} for the known input $\mathbf{x}^{N+1}$. We start with computation of covariance matrix $\mathbf{C}_N$ of size $N \times N$ with components:

$$c_{nm} = k(\mathbf{x}^n, \mathbf{x}^m) + \frac{1}{2\sigma^2}\delta_{nm}. \tag{185}$$

The conditional distribution $p(t^{N+1}|\mathbf{t})$ can be found as Gaussian distribution

$$p(t^{N+1}|\mathbf{x}^{N+1}, \mathscr{D}_N) = \mathscr{N}(t^{N+1}|m_{N+1}, \sigma^2_{N+1}), \tag{186}$$

where the mean and variance are computed from the Gaussian pd, see Bishop (2006), pp. 307-308:

$$m_{N+1}(\mathbf{x}^{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t}, \tag{187}$$

$$\sigma^2_{N+1} = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \tag{188}$$

The vector $\mathbf{k}$ and scalar c are components of the covariance matrix $\mathbf{C}_{N+1}$:

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & c \end{bmatrix}, \tag{189}$$

where:

$$\mathbf{k}^{\mathrm{T}} = \{k(\mathbf{x}^n, \mathbf{x}^{N+1})\}_{n=1}^N, \quad c = k(\mathbf{x}^{N+1}, \mathbf{x}^{N+1}) + \frac{1}{2\sigma^2}, \tag{190}$$

which enables computation of Gaussian joint distribution

$$p(\mathbf{t}^{N+1}) = \mathscr{N}(\mathbf{t}^{N+1}|\mathbf{0}, \mathbf{C}_{N+1}), \tag{191}$$

for $\mathbf{t}^{N+1} = \{t^n\}_{n=1}^{N+1}$.

It is worth mentioning that in the GP method weight parameters are not used. The main problem of the approach is computation of the inverse matrix $\mathbf{C}_N$ which requires $O(N^3)$ computations. A vector-matrix multiplication needs $O(N^2)$ computations. It is clear that in case of large data sets the curse of dimensionality appears.

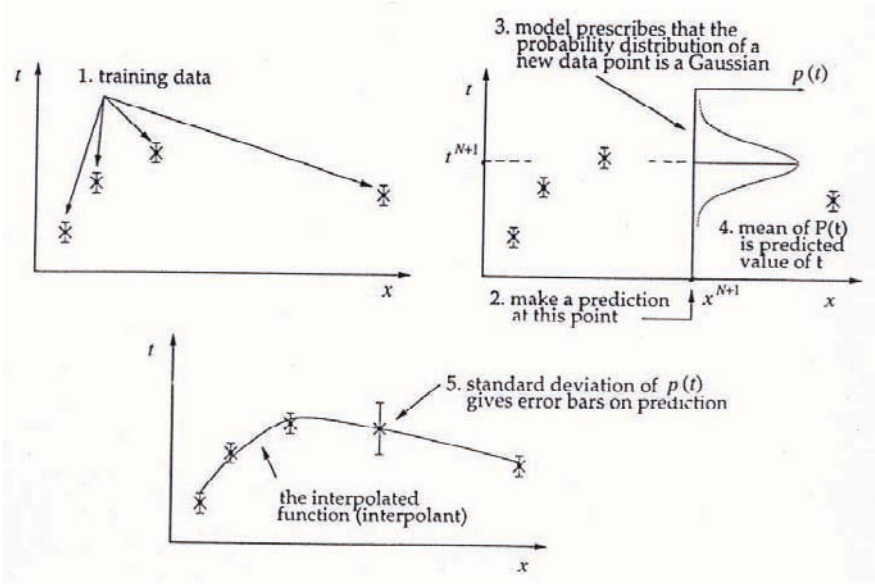


Figure 26. Schematic description of GP method for one-dimensional regression problem

By contrast, the application of interpolation RBFN needs the numbers of corresponding computations $O(W^3)$ and $O(W^2)$, respectively for $W = K + 1$, where K is the number of RBFs. However, an advantage of the GP framework is that we can take into account covariance functions which could only be expressed in terms of infinite number of RBFs. More information on the GP method can be found in Bishop (2006), pp. 303-320.

5.10 Bayesian inference and ANNs

Let us sum up this Section by writing relations between ANNs and Bayesian inference, which are, in general, completed of the learning and prediction parts, see Table 2, where the predicted pattern is marked as $(\mathbf{x}^{N+1}, t^{N+1})$.

It was proved that the Bayesian Maximum Likelihood and corresponding SNN/ML fully correspond to searching of the minimum in the Penalized Least Square error method and application of the classical, deterministic networks SNNs.

The Bayesian Maximum A Posterior approach can be used for improving learning of deterministic network SSN/MAP by means of error minimization of the penalized cost function.

Simple BNN is based on "a more" Bayesian approach. This means that hyperparameters are deterministic and their values can be improved in an iterative way.

Table 2. ANN framework, learning quantities and prediction functions

ANN and Bayesian frameworks	Learned quantities	Prediction functions
1. SNN/ML or SNN/MAP	$\mathbf{w}_{\mathrm{ML}}$ or $\mathbf{w}_{\mathrm{MAP}}$	$y(\mathbf{x}^*; \mathbf{w}_{\mathrm{ML}})$ or $y(\mathbf{x}^*; \mathbf{w}_{\mathrm{MAP}})$
2. S-BNN (Simple NN)	$p(\mathbf{w}\mid \mathcal{D}, \alpha_{\mathrm{opt}}, \sigma^2_{\mathrm{opt}})$	$p(t^{N+1}\mid \mathbf{x}^*, \mathcal{D}, \alpha_{\mathrm{opt}}, \sigma^2_{\mathrm{opt}})$
3. T-BNN (True NN)	$p(\mathbf{w}\mid \mathcal{D}, \alpha, \sigma^2)$, $\alpha_{\mathrm{MP}} = \mathcal{E}[\alpha(\mathbf{w}_{\mathrm{MAP}})]$, $\beta_{\mathrm{MP}} = \mathcal{E}[\beta(\mathbf{w}_{\mathrm{MAP}})]$	$p(t^{N+1}\mid \mathbf{x}^*, \mathcal{D}, \alpha_{\mathrm{MP}}, \sigma_{\mathrm{MP}}{}^2)$
4. GP-BNN (Gauss Process NN)	$\mathbf{C}_N{}^{-1}(\mathbf{X}^N), \sigma^2_{N+1}, \boldsymbol{\theta}$	$p(t^{N+1}\mid \mathbf{x}^{N+1}, \mathbf{t}, \mathbf{C}_{N+1}{}^{-1}, \boldsymbol{\theta}, \sigma^2_{N+1}) =$ $\mathcal{N}(\mathbf{x}^{N+1}\mid m_{N+1}(\mathbf{x}^{N+1}, \mathbf{C}_N{}^{-1}, \mathbf{t}, \boldsymbol{\theta}, \sigma^2_{N+1})$

In this approach numerical methods for the integration of pds are introduced, e.g. Hybrid Monte Carlo method.

In the True BNN the hyperparameters α_{MP}, σ^2_{MP} are assumed to be random values and they are computed as variables coupled with the weight vector $\mathbf{w}$ and data set $\mathcal{D}$.

The application of the Gaussian Process in the Bayesian inference is based on the computation of the total, inverse covariance matrix $\mathbf{C}_N^{-1}(\mathbf{X}^N, \sigma^2)$ in the input space. In this approach we resign from the weight parameter vector $\mathbf{w}$ and instead, in kernels, we introduce the parameter $\boldsymbol{\theta}$, cf. (183). The most important point is that we assume the Gaussian posterior in which the mean m_{N+1} and variance σ^2_{N+1} parameters are computed, cf. (187) and (188).

In Table 2, possibilities of using various approximations are shown and attention is also focused on the application of the Gaussian Process approach that is now at the research top of the Bayesian analysis.

6 Applications of ANNs to the Analysis of Selected Engineering Problems

The Section is devoted to discussion of several engineering problems, selected from the viewpoint of illustration of the topics discussed in previous Sections of the Chapter. All the presented problems have been developed in research conducted in recent years at the Institute of Computer Methods in Civil Engineering (now Institute for Computational Civil Engineering) of the Cracow University of Technology, Poland, see Waszczyszyn (2006).

6.1 Dynamics of buildings subjected to paraseismic excitations

In the recent 15 years a great deal of attention has been paid at the Cracow University of Technology, Poland to projects related to research on vibrations of real buildings. These buildings were subjected to so-called paraseismic excitations caused by mining tremors, explosions in quarries, traffic on motorways. A great number of problems and the evidence collected in the corresponding data banks have also been explored in the development of neural network engineering applications in Poland during the last ten years, see Waszczyszyn (2006).

In what follows we focus only on the data obtained during monitoring of vibrations of a five-storey, prefabricated building (cf. Figure 27), see book by Kuźniar (2004) and a review by Kuźniar and Waszczyszyn (2007). The corresponding problem was an excellent playground to learn and develop ANNs, oriented on the analysis of various problems of structural dynamics and experimental mechanics.

Simulation of fundamental periods of natural vibration. Natural periods, vibration damping and mode shapes of natural vibrations characterize dynamic properties of structures. We concentrate only on the fundamental periods of natural vibrations which were used in simple expert systems developed for the evaluation of the technical state of buildings, subjected to mining tremors and explosions in nearby mines or quarries, cf. Ciesielski et al. (1992).

Referring to data discussed by Kuźniar (2004) we focus on a group of 31 monitored, prefabricated buildings of different types. Natural vibrations are excited by propagated surface seismic waves so the full-scaled measured accelerograms were used to compute 31 target patterns as T_1 [sec.] periods of vibrations (the Fast Fourier transformation procedure was applied). On the base of extensive research, see references in Kuźniar (2004), the following input and output variables were selected:

$$\mathbf{x}_{(4\times1)} = \{C_z, b, s, r\}, \quad y = \{T_1\}, \tag{192}$$

where: C_z – ratio of vertical unit base pressure for elastic strain; b – building dimension along the vibration direction (longitudinal or transversal); $s = \sum_i EI_i/a$, $r = \sum_i GA_i/a$ – equivalent building and shear stiffnesses of the i-th internal walls in the building plan, cf. Figure 27a.

The components of the input vector in $(192)_1$ are arranged according to their importance. This means that we can omit the input s but never C_z. That is why in the analysis the following inputs were also included:

$$\mathbf{x}_{(3\times1)} = \{C_z, b, s\}, \quad \mathbf{x}_{(2\times1)} = \{C_z, b\}, \quad \mathbf{x}_{(1\times1)} = \{C_z\}. \tag{193}$$

All the variables listed in (193) were rescaled to the range (0,1).

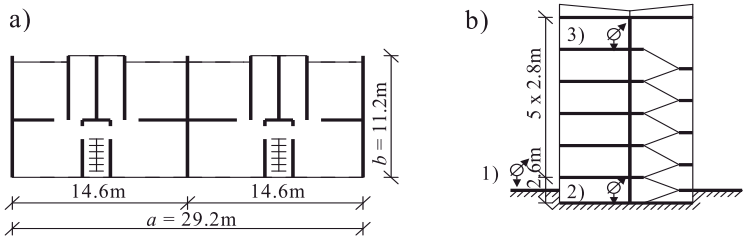


Figure 27. Prefabricated five-storey building of WK-70 type: a) Plan, b) Sectional elevation and points of vibration measurements

In paper by Ciesielski et al. (1992) the following empirical formula was proposed:

$$T_1 = \frac{0.98}{\sqrt[3]{C_z}}, \tag{194}$$

where C_z should be substituted from the range [50-300] MPa/m without rescaling C_z.

A similar formula was derived by Kuźniar et al. (2000):

$$T_1 = \frac{1.2}{\sqrt[3]{C_z + 0.003(s+r)/b}}. \tag{195}$$

Because of a small number of patterns $P = 31$ only simple neural networks were applied for all the used inputs. From among many tested networks only FLNNs networks of architectures 4-4-1, 2-3-1 and 1-2-1 as well as the number of their parameters *NNP* are listed in Table 3.

The total set of patterns used was 100 times randomly split into the training and testing sets, completed of $L = 25$ and $T = 6$ patterns. In Table 3 the average errors are shown for 100 training processes. The standard deviation $\sigma(P)$ and coefficient of correlation $r(P)$ were computed for the whole set of non-scaled patterns. The *MSE* and *ARE* errors, defined in Appendix as (A1) and (A3), were computed for the input variables rescaled to the range (0, 1). The output variable was from the range [0.155, 0.294] sec. so it was not rescaled.

Looking at the results we can see that network (1): 4-4-1 is superior to network (2): 2-3-1. Both neural networks are much better than empirical formulae (194) and (195).

Next the PCA method was applied to the reduction of number of inputs. Applying PCA, described in Point 3.4, the eigenvalues and eigenvectors λ_j, $\mathbf{q}_j$ were computed for a covariance matrix $\mathbf{S}_{(4\times4)}$. They are presented in Table 4.

As can be seen the errors listed in Table 3 for the more complex network (1) of structure 4-4-1 are comparable with errors which were obtained by networks

Table 3. Errors and statistical parameters for different input variables and different architectures of FLNNs

Nos of models	Input para-meters	FLNN	*NNP*	*MSE*×10^3		*ARE* [%]		$\sigma(P)$	$r(P)$
				L	T	L	T		
(1)	C_z, b, s, r	4-4-1	25	0.08	0.32	3.0	5.8	0.011	0.964
(2)	C_z, b	2-3-1	13	0.25	1.20	6.5	12.5	0.020	0.873
(3)	ξ_1, ξ_2	2-3-1	13	0.07	0.23	3.5	6.5	0.010	0.972
(4)	ξ_1	1-2-1	7	0.19	0.70	7.0	5.3	0.017	0.901
(194)						12.8		0.045	0.768
(195)						11.9		0.029	0.793
(196)						8.9		0.024	0.811

Table 4. Eigenvalues λ_j, relative eigenvalues m_j and eigenvectors $\mathbf{q}_j$ of covariance matrix **S**

No. of PC j	Eigenvalues λ_j	Coeff. (3.40) m_j(%)	Eigenvectors $\mathbf{q}_j$
1	32.029	0.9516 (95.1)	{ 0.0852, 0.0589, 0.0019, 0.9946}
2	1.548	0.0460 (4.6)	{–0.3472, –0.9339, –0.0072, 0.0850 }
3	0.081	0.0024 (0.2)	{ 0.9339, –0.3526, 0.0070, –0.0592 }
4	0.001	0.0000 (0.0)	{ 0.0092, 0.0044, –0.9999, 0.0009 }

(3) with only two PC inputs. The same concerns simple networks (2) and (4) with one PC input. It is interesting that the architectures 2-3-1 give better results for PC inputs.

In Figure 28 there are shown distributions of measured (target) fundamental periods $t^n = T^n_{1\text{meas}}$ [sec.] vs. the periods $y^n = T^n_{1\text{PC}}$, predicted by the networks 1-2-1 and 2-3-1 with the PC single input and double inputs. It is visible that for two PC inputs ξ_1 and ξ_2, nearly all the patterns predicted by network 2-3-1 are placed within the relative error cone $Bep = 5\%$. It is defined as an area of the relative absolute errors of the network output $|1 - y^n/t^n| \times 100\% \leq |Bep|$.

On the base of PCA a new empirical formula was derived using only the first principle component ξ_1, see Kuźniar and Waszczyszyn (2007). In Figure 29 there are shown the pattern points for which the following cubic regressive curve was computed by means of the LS method:

$$T_1 = 0.238 + 0.08\bar{\xi}_1 - 0.1165\bar{\xi}_1^2 + 0.03\bar{\xi}_1^3, \tag{196}$$

where

$$\bar{\xi}_1 = (0.0852C_z + 0.0589b + 0.0019s + 0.9946r)/100. \tag{197}$$

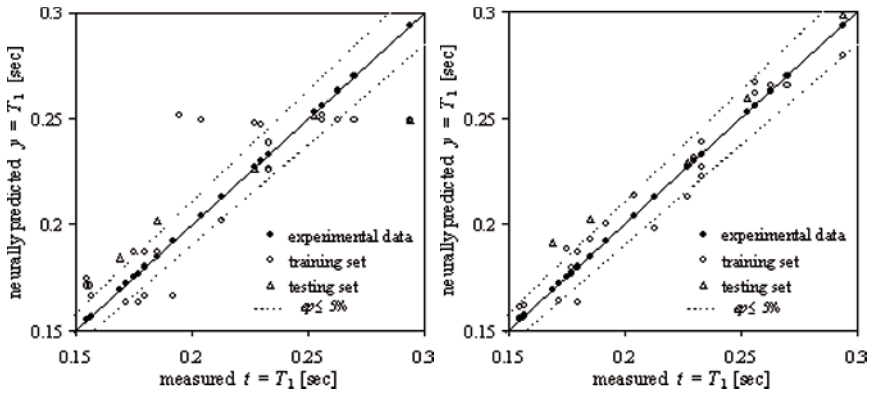


Figure 28. Experimental vs. neurally predicted fundamental vibration periods computed by networks: a) 1-2-1 and $x = \xi_1$, b) 2-3-1 and $\mathbf{x} = \{\xi_1, \xi_2\}$

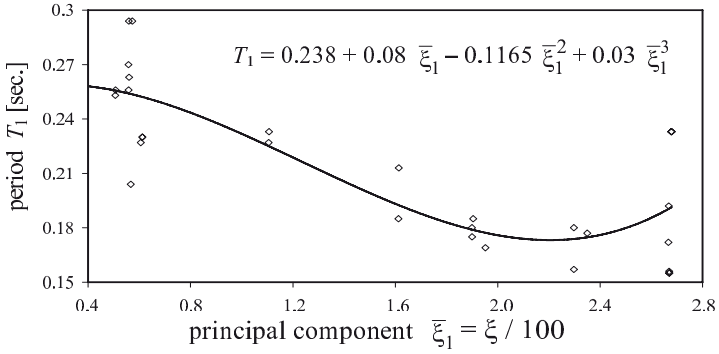


Figure 29. Fundamental period T_1 vs. the first principal component $\bar{\xi}_1$

In Table 3, the errors of prediction of the fundamental period T_1 by formula (196) are listed. The results of prediction of $ARE(P), \sigma(P)$ and $r(P)$ are better estimated by formula (196) than those by empirical formulae (194) and (195).

Simulation of DRS for soil-structure interaction problems. Response spectra are often applied in structural design and for determining dynamic resistance of existing buildings, cf. e.g. Eurocode 8 (2003). The response spectrum is defined by means of the motion of a 1DOF oscillator starting from the equation:

$$\ddot{x} + 2\xi\omega_i\dot{x} + \omega_i^2 x = -a_g(t), \tag{198}$$

where: $\omega_i = 2\pi f_i = 2\pi/T_i$ – angular frequency, $T_i = 1/f_i$ – period of vibrations,

ξ – damping coefficient, $a_g(t)$ – excitation corresponding to ground acceleration. Knowing the measured record of accelerations, we can digitize it and use to compute displacements $x_j = x(t_j)$ for fixed values of frequencies f_i or periods of vibration T_i at an assumed damping coefficient ξ.

Then the *Displacement Response Spectrum* (DRS) can be computed as a function which maps the natural periods of oscillators into the maximal values of their displacement response:

$$Sd(T_i) = \max_j |x(t_j; T_i, \xi)|. \tag{199}$$

In Figure 30 an example of the measured acceleration record and the corresponding computed DRS are shown.

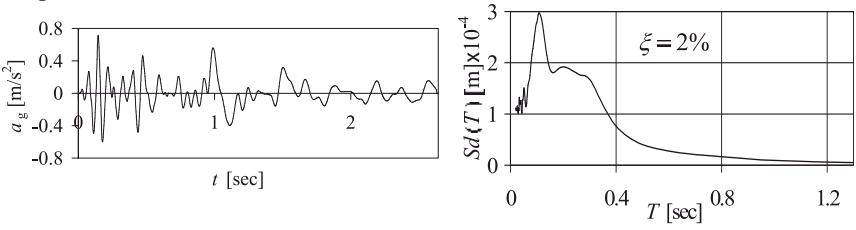


Figure 30. Measured accelerogram and corresponding computed DRS

The problem of soil-structure interaction is to find a response spectrum DRSb for the excitation spectrum DRSg, i.e. to make the mapping DRSg $\rightarrow$ DRSb. In this relation DRSg is a spectrum computed on the basis of measurements made on the ground level outside the building (measurement point 1 in Figure 27b) and DRBb is a spectrum for measurements performed inside the building at the basement level (measurement point 2). It is rather a difficult task to compute the motion of the structure by, for instance FE models. In case of real structures and application of FEM there are many serious problems related to modelling of boundary conditions, connections between structural elements, material relationships, etc. A number of these issues can be overcome due to ANNs applications, cf. references in Kuźniar and Waszczyszyn (2007).

The problem discussed is related to medium height (5-storey), prefabricated buildings in Legnica - Głogów Copperfield, Poland. The buildings were subjected to paraseismic excitations caused by firings of explosives. Ten accelerograms were randomly selected from among those measured at monitored buildings. The corresponding discrete values Sdg_k and Sdb_k were computed for $k = 1, \ldots, 198$ periods of natural vibrations corresponding to $T_k \in [0.02, 1.3]$ sec.

Using static approach and the temporal window method, the following input and output vectors were adopted, see Kuźniar (2003):

$$x_{(6\times 1)} = \{Sdg_{k-2}, Sdg_{k-1}, Sdg_k, Sdg_{k+1}, Sdg_{k+2}, T_k\}, \quad y = Sdb_k, \tag{200}$$

where: $k = 3, ..., 196$ – index of discrete time for successive vibration periods T_i.

The set of 10 pairs of DRS values $\{\{Sdg_k\}_{k=1}^{198}, \{Sdb_k\}_{k=3}^{196}\}$ was randomly split into an equal number of 5 training and 5 testing sets, respectively. The corresponding numbers of training and testing patterns were $L = T = P/2 = 5 \times 198 = 990$. These patterns were used for training of FLNN shown in Figure 2.4a. The Rprop learning method was used (cf. Waszczyszyn (1999), p.20) in the applied SNNS simulator, see Zell et al. (1994). After the cross-validation method was used the network BPNN: 6-5-1 was designed (the acronim BPNN was used in papers by Kuźniar (2003) instead of FLNN). The errors of this network of training and testing are shown in Table 5 (acronym BPNN was used in papers by Kuźniar instead of FLNN applied in this paper).

Table 5. Errors of training and testing for neural predictions of DRSb

Autoregressive time-delay input		Learning method	$MSE(V) \times 10^4$		ARE[%]		
			L	T	L	T	$r(T)$
2-5-1	FLNN	Rprop	13.7	12.5	13.7	13.3	0.879
	FLNN	DEKF	3.9	6.5	10.8	10.4	0.958
	FLNN	Conj grad	3.68	6.51	3.63	3.79	0.980
	FLNN	MAP	3.70	6.51	3.60	3.79	0.980
	BNN	Simple	3.69	6.52	3.58	3.71	0.981
	BNN	True	3.66	6.44	3.57	3.69	0.981
2-4-1	FLNN	Conj grad.	3.70	6.60	3.74	3.79	0.980
	FLNN	MAP	3.69	6.60	3.70	3.88	0.980
	BNN	Simple	3.68	6.61	3.65	3.97	0.980
	BNN	True	3.57	11.86	3.83	5.80	0.965
2-3-1	FLNN	Conj grad.	3.63	6.59	3.59	3.81	0.980
	FLNN	MAP	3.66	6.48	3.61	3.70	0.980
	BNN	Simple	3.67	6.51	3.67	3.79	0.980
	BNN	True	3.68	6.56	3.74	3.85	0.980

The graphics of two selected DRS l#1 and DRS t#3, used for the network training and testing are shown in Figure 31. The predicted FLNN, corresponding to the neural network BPNN: 6-5-1 learnt by the Rprop method are marked as BPNN DRSb.

Application of Kalman filtering. The results quoted below were obtained in Krok's PhD dissertation, Krok (2007) and were quoted in the review paper by Waszczyszyn and Ziemiański (2005). The following sequential values were adopted as input and output variables:

$$x_{(2\times 1)} = \{Sdg_{k-1}, Sdb_{k-1}\}, \quad y = Sdb_k. \tag{201}$$

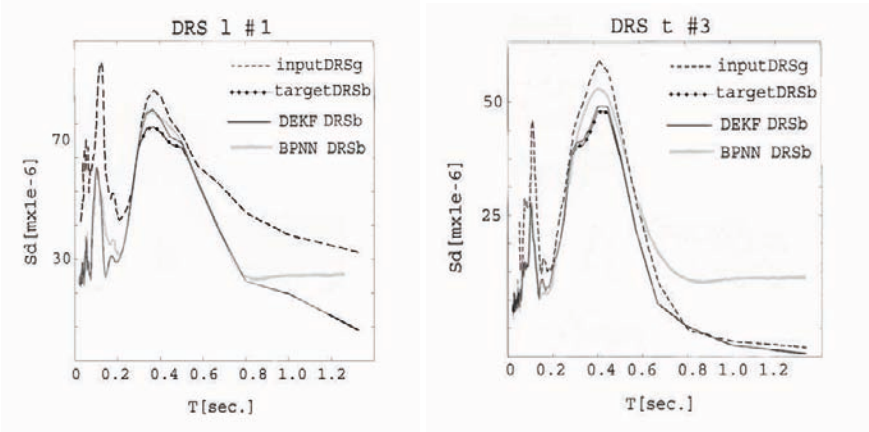


Figure 31. Displacement learning and testing spectra DRS l#1 and DRS t#3, related to measured spectra at ground and basement levels (input DRSg and target DRSb), vs. spectra computed by Kalman filtering (DEKF DRSb) and Rprop learning method (BPNN DRSb)

where: Sdg_{k-1} – value of DRSg at the ground level for discrete time $k-1$; Sdb_{k-1}, Sdb_k – values of DRSb at the basement inside the building for $k-1$ and k discrete times $k = 2,3,...,198$.

The autoregressive time-delay input Sdb_{k-1} was assumed as a variable well fitting the character of the Kalman filtering method. Preliminary computations were performed using two types of neural networks, discussed in Point 4.2.1, i.e. the feed-forward network FLNN, and recurrent network RLNN. After introductory computations, it was stated that FLNN was superior (contrary to another paper written also on the simulation of Response Spectra, see Krok and Waszczyszyn (2007)). The training was performed by author's procedures written in the MATLAB language related to the simulator Neural Network Toolbox for Use with Matlab, see Demuth and Beale (1998).

On the basis of numerical experiments the following functions of the Gaussian noises were found:

$$\mathbf{Q}(k) = 0.01\exp(-(s-1)/50)\mathbf{I}, \quad R(k) = 7\exp(-(s-1)/50), \tag{202}$$

where: $\mathbf{I}$ – unit matrix of dimension (3×3) for the $j = 1,2,\ldots,5$ neurons of the hidden layer and (6×6) matrix for the output; s – number of the epoch in training process. The stopping criterion was established with respect to the fixed number of epochs S corresponding to the testing error $MSE(T) < \varepsilon$. After introductory computations the stopping criterion was related to $S = 500$ assuming $\varepsilon_{\text{adm}} = 1\times 10^{-4}$.

Training of network FLNN: 2-5-1 (in Figures 31 the corresponding results are marked as BPNN DRSb) was performed using the algorithm DEKF as a learning method. The same training sets were used as those discussed in the previous point

where network BPNN: 6-5-1 was trained by means of the Rprop learning method. Training and testing errors are listed in Table 5 and corresponding graphics of selected DRS are shown in Figure 31.

Bayesian approaches. The results quoted below are mainly taken from paper by Waszczyszyn and Słoński (2006). They have recently been completed by Słoński with respect to the design of optimal neural networks.

In order to apply the criterion MML (Maximum of Marginal Likelihood) for the total number of ten DRS the curve $\ln p(\mathscr{D}|\alpha,\beta)$ was first plotted using formula (154). In this formula the number of weights W corresponds to the number of hidden neurons H in the network FLNN: 2-H-1. In the introductory computation the following values of hyperparameters were found $\alpha = 2.58$ and $\beta = 2710$. These values were kept constant during the computation. Then Type-II ML approximation, described in short in Point 5.8, was used. Starting from the Gaussian pd, the following optimal values of hyperparameters $\alpha_{\text{opt}} = 2.53$ and $\beta_{\text{opt}} = 1980$ were computed.

In Figure 32 the plots of $\ln Mal(H;\mathscr{L})$ are shown for the number of training data sets of patterns $L = 5$ and 10. These sets fully correspond to sets used in the computations carried out in paper by Waszczyszyn and Słoński (2006). From the plots we can conclude that the optimal number of neurons equals $H_{\text{opt}}^{\text{Mal}} = 4$ for the total number of sets $L = P = 10$ and $H_{\text{opt}}^{\text{Mal}} = 3$ only if the training $L = 5$ sets are applied.

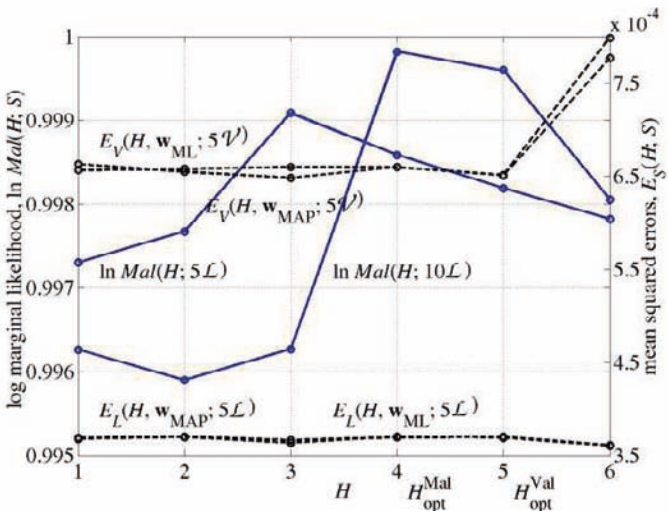


Figure 32. Marginal Likelihood Curves $\ln Mal(H;\mathscr{D})$ computed for $L = 5$ and $L = P = 10$ sets of patterns and validation error curve $E_{\mathscr{V}}(H,\mathbf{w};5\mathscr{V})$ for the network FLNN: 2-H-1

In the same figure the validation plots $E_{\mathscr{V}}(H,\mathbf{w};5\mathscr{V})$ are drawn for 5 validation sets $\mathscr{V}$ and various vectors of weights. The vector $\mathbf{w}_{\mathrm{ML}}$ was computed by means of standard NN using the conjugate gradient learning method but without the penalized weight-delay function (this corresponds to the application of SNN/ML method listed in Table 2). The vector $\mathbf{w}_{\mathrm{MAP}}$ was computed by the Bayesian network SNN/MAP in which the method of Maximum A Posterior was explored. It is visible that from the cross-validation method point of view the optimal value (minimum of the function $E_{\mathscr{V}}(H,\mathbf{w};5\mathscr{V})$ corresponds to $H_{\mathrm{opt}}^{\mathrm{Val}}=5$ but the values $H=2-4$ could also be acceptable. That is why the computations were carried out for the hidden neuron numbers $H=3-5$ and the obtained errors are listed in Table 5.

Results of two Bayesian approaches are presented in Table 5. These approaches are defined according to Table 2. The Simple Bayesian network S-BNN is based on the deterministic values of hyperparameters, which were iteratively updated. In the analyzed network the initial values of those parameters were $\alpha_{\mathrm{in}}=2.58$ and $\beta_{\mathrm{in}}=1/\sigma_{\mathrm{in}}^2=2710$ were updated to values $\alpha_{\mathrm{opt}}=2.58$ and $\beta_{\mathrm{opt}}=1/\sigma_{\mathrm{opt}}^2=1980$. The NETLAB simulator (see Nabney (2004)) and the FBM procedures (see Neal (2004)), were used. The Hybrid Monte Carlo method of numerical integration (without persistence) and Gibbs sampling were applied.

The errors related to application of all the networks discussed above are shown in Table 5. Looking at them we can conclude that results obtained by S-BNNs and T-BNNs are very close to each other. Moreover, the complexity of neural networks does not affect results of computations. Such a conclusion is valid only for the data sets used. The application of 10 records seems to be too small to draw more general conclusions.

Plots of target and Bayesian inference predictions are shown in Figure 33 for the same DRSs as those in Figure 31, taken from paper by Waszczyszyn and Słoński (2006). Besides the means also $2\sigma_{\mathrm{t}}$ error bounds are plotted. It is visible that an excellent fitting of these curves to target means of DRSb takes place.

6.2 Analysis of hysteresis loops for a super-conducting cable

The Kalman filtering was applied in multilayer neural networks for the analysis of hysteresis loops which occur in materials and structural elements subjected to cyclic loading. This approach was developed in Krok's PhD dissertation, Krok (2007), in which hysteresis loops for concrete and steel specimens were simulated and predicted. A part of dissertation, published by Krok in paper Krok (2006), was devoted to a special problem related to the analysis of hysteresis loops in a super-conductor cable, placed in a cryogenic box and subjected to cyclic pressure.

The superconductor was designed for the International Experimental Reactor,

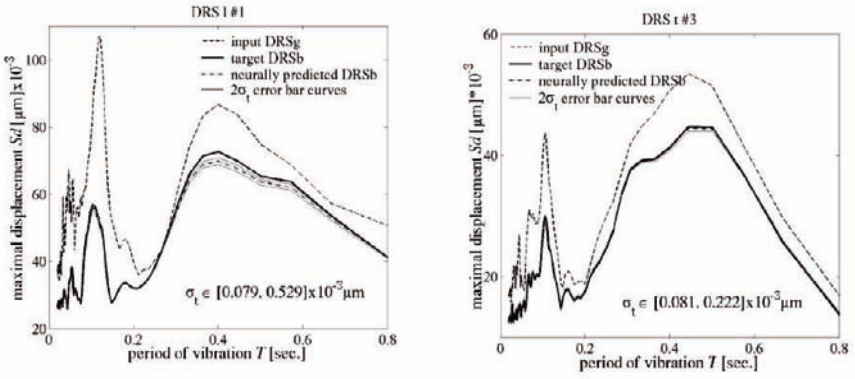


Figure 33. Displacement Response Spectra, target and computed by S-BNN for: a) set l#1, b) testing set t#3

(ITER), see the website www.iter.org. The scheme of cable cross-section and mechanical stand-up are shown in Figure 34. Measured hysteresis loops are shown in Figure 35a. These Figures were taken from the report by Nijihuis et al. (1998). In Figure 35a, nine representative cycles (with Nos $1, \ldots, 6, 11, 21$ and 38) were plotted, from among 38 cycles carried-out.

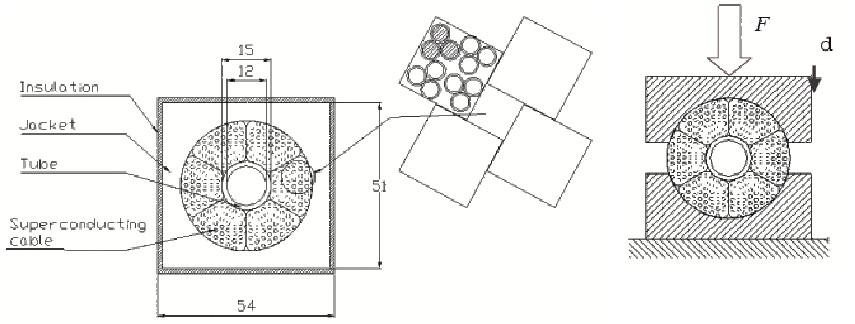


Figure 34. a) Cable blocks, steel jacket and cooling tube, b) Tested cable

These experimental results, taken from the above quoted report, were analysed in many papers, see references in book by Lefik (2005). The standard, multilayered neural networks were also applied, see Lefik and Schrefler (2002). Results of an extensive numerical analysis were described by Krok (2007). She focused on selection of input variables and applied the Kalman Filtering algorithm DEKF for learning of networks FLNN and RFLN, see Figure 13.

Following the approach by Lefik and Schrefler (2002), the hysteresis loops can be plotted using discrete time k, numbering a sequence of $K = 244$ points, marked

in Figure 36. After such a mapping onto time domain the Kalman filtering can be applied in a similar manner as shown in Point 6.1. The patterns were defined to pairs (F_k, d_k), where: F_k, d_k – force and transversal displacement at the k-th instant.

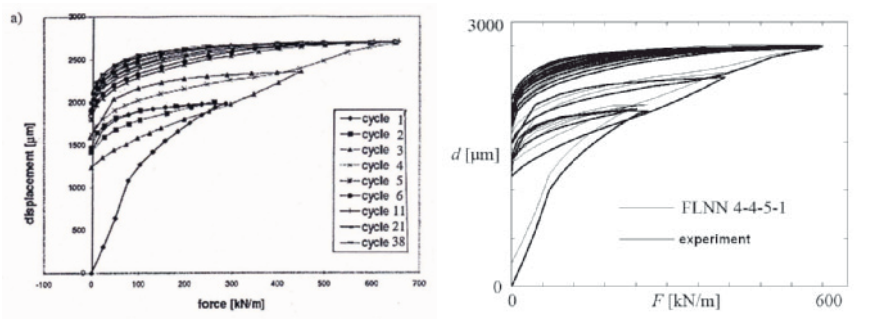


Figure 35. Hysteresis loops taken from: a) Laboratory testing, b) Neural simulation

Following the paper by Lefik and Shrefler quoted above, three selections of training and testing patterns were adopted:

a) 25% or 50% of all 244 patterns were randomly selected for testing;
b) the first seven loops containing $L = 180$ patterns were used for training and three final loops with $T = 64$ patterns served for testing;
c) two first loops containing $T = 35$ patterns were used for testing and the remaining seven loops with $L = 209$ patterns were explored for training.

The starting inputs and output were completed from the data pairs:

$$\mathbf{x}_{(3\times1)} = \{F_k, F_{k+1}, d_k\}, \quad y = \{d_{k+1}\}. \tag{203}$$

In the input vector the time-delay, autoregressive variable d_k was applied. All the values were rescaled to the range (0,1).

In Krok's paper various combinations of four inputs were analysed. From among them what superior was the following input vector:

$$\mathbf{x}_{(4\times1)} = \{F_{k+1}, d_k, 1 - k/244, l(k)\}. \tag{204}$$

Besides conserving of the time-delay input d_k, a variable $l(k)$ was introduced. It corresponds to the subsequent values: 1-1/N(1), 1-2/N(2), 1-$N(l)$/$N(l)$, where $N(l)$ is the total number of measurements in the l-th loop. It is worth mentioning that the 4-th input parameter of a loop switching character was also introduced by Lefik and Schrefler (2002).

In Table 6 *MSE* errors are shown for selected networks, taken from paper by Krok (2006). Results shown in the Table follow three cases of the testing patterns

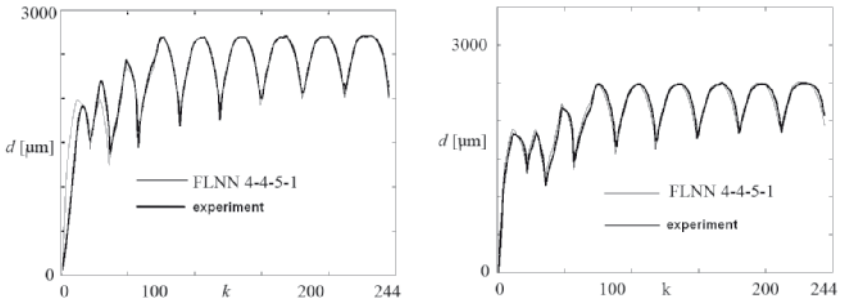


Figure 36. Extended hysteresis loops for testing patterns: a) 25% random selection, b) two first loops

selection. Only two architectures of networks were selected, i.e. 3-15-1 and 4-4-5-1, corresponding to inputs (203) and (204). The numbering of study cases corresponds to that in Krok's paper.

Table 6. Testing errors for different networks with Kalman filtering applied to the analysis of hysteresis loops in super-conducting cable

Number of study cases		ANN	Testing	MSE×10^3	
				L	T
a)	1(25%)	FLNN: 4-4-5-1	T = 0.25 P	0.27	0.37
	1(50%)	FLNN: 4-4-5-1	T = 0.5 P	0.36	0.30
b)	2	RLNN: 3-15-1	Two	0.86	0.35
	3	FLNN: 3-15-1	final	0.88	0.46
	8	FLNN: 4-4-5-1	loops	0.39	0.35
c)	10	RLNN: 3-15-1	Two	0.53	2.03
	11	FLNN: 3-15-1	first	0.67	2.09
	15	FLNN: 4-4-5-1	loops	0.67	1.92

Two hidden layer network 4-4-5-1 has the number of network parameter NNP = 51, whereas the network 3-15-1 has NNP = 76 for FLNN and NNP = 92 for RLNN. The testing errors are comparable for cases a) and b) but the $MSE(T)$ error for c) is significantly higher. The influence of the testing set selection is visible in Figure 36. The testing fitting curves are very close to the target curves in cases a) and b). In case c) the predicted curve for the first three loops does not fit well the target curve shown in Figure 36b.

6.3 Bayesian inference in the microstructure analysis

A great deal of attention is devoted to multiscale analysis of heterogeneous materials. Computational Homogenization (CH) has been quite recently proposed as a method based on the advanced computer hard- and software, see e.g. PhD dissertations by Kouznetsova (2002) and Kaczmarczyk (2006). CH is a two levels approach in which refined mechanical formulation is applied on the microscale level and the implicit formulation is used on the macroscale level of observation. The microlevel is represented by RVE (Representative Volume Element) which is placed in an integration point of the Finite Element, representative for the macrolevel, see Figure 37.

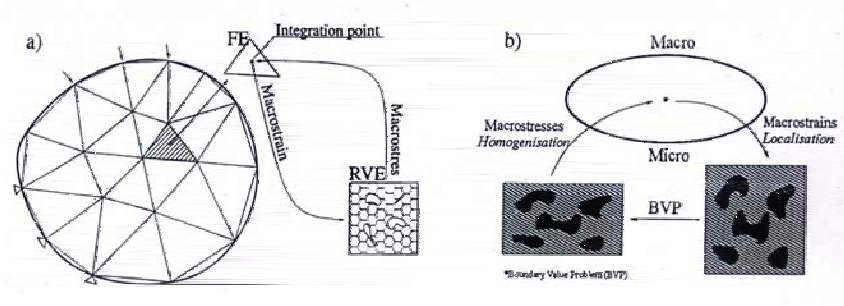


Figure 37. a) Main scheme of Computational Homogenization, b) Micro-macro transition

The analysis of RVE plays a basic role in CH. In this area the basic question concerns the identification of characteristics of RVE. Among them the characteristic length L of RVE should be defined and, moreover, its value should be estimated. The paper by Kaczmarczyk and Waszczyszyn (2007) focused on computation of the characteristic length L of a plane RVE shown in Figures 38, 40 applying FEM and Bayesian inference explored in the T- BNN network.

The characteristic length L reflects the micro-macro scales interaction, mechanical properties of microstructure ingredients and nonlinear deformations on the micro-macro levels. These effects can be experimentally tested by means of the indentation test, see Figure 39a. A rigid indenter is pressed in a deformable matrix, so a great stress concentration and deformation localization occur in the vicinity of indenter application. These phenomena need a great increase of the FE mesh density in the stress and deformation regions, see Figure 39b.

The plain strain Boundary-Value-Problems (BVPs) were formulated on the micro- and macro levels. It was assumed that the aluminium matrix has elasto-plastic properties and the inclusions are made of elastic silicon (their mechanical parameters are shown in Figure 38). The second order continuum with micro-strains was assumed to analyze the RVE deformation. The displacement boundary

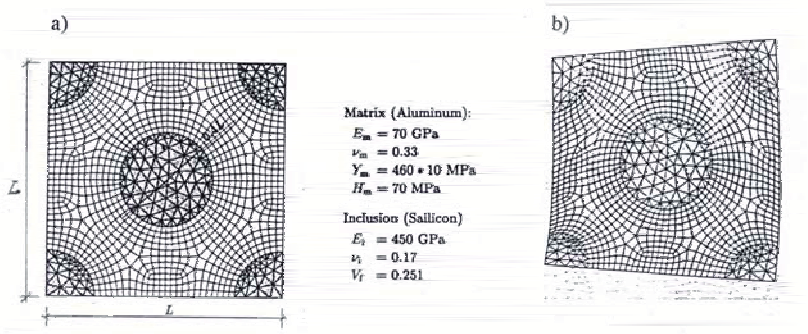


Figure 38. a) Discretization of RVE, b) Deformation of RVE in the vicinity of indenter

constraints for RVE were applied and friction between the indenter and the deformable continuum was neglected.

A sequence of BVPs was analysed performing 34 incremental steps Δh to compute the equilibrium paths $F(h)$, where: F, h – force and vertical displacements shown in Figure 39a. A deformation of RVE in the vicinity of the indenter application is shown in Figure 38b. The equilibrium paths were computed for three, fixed in advance, values of the length $L = 0.001, 0.002$ and 0.004 mm.

The vector of sequential input variables was composed of 105 variables:

$$\mathbf{x}_{(105\times 1)} = \{h_1, h_2, \ldots, h_{35}, F_1, F_2, \ldots, F_{35}, s_1, s_2, \ldots, s_{35}\}, \tag{205}$$

where s is the width of indenter adherence to deformable continuum, see Figure 39a.

Prior distribution of the characteristic length $p(L)$ was estimated by Gamma pdf assuming in (178) $\alpha = L$, $a = 2$ and $b = 0.5$. It is worth noting that this prior has no reference to any data and prior estimation about what the true relationship might be.

Assuming three input variables, i.e. $\mathbf{x} = \{h, F, s\}$, a set of 344 pseudo-experiments was formulated performing 34 incremental steps corresponding to input vector (205). In order to diminish the dimensionality of the input space the PCA method was applied. This analysis gave the following eigenvalues

$$\lambda_1 = 0.055, \quad \lambda_2 = 8.69 \times 10^{-6}, \quad \lambda_3 = 4.83 \times 10^{-6}. \tag{206}$$

In the following analysis only one PC, i.e. $\xi_1(h, F, s)$ was conserved.

The Bayesian neural network T-BNN: 1-16-1 was applied and randomly selected, the training and testing sets were used, composed of $L = 200$ and $T = 144$ patterns, respectively. The training process, corresponding to the Bayesian inference, was preceded according to that described in Point 6.1. In Figure 40 the

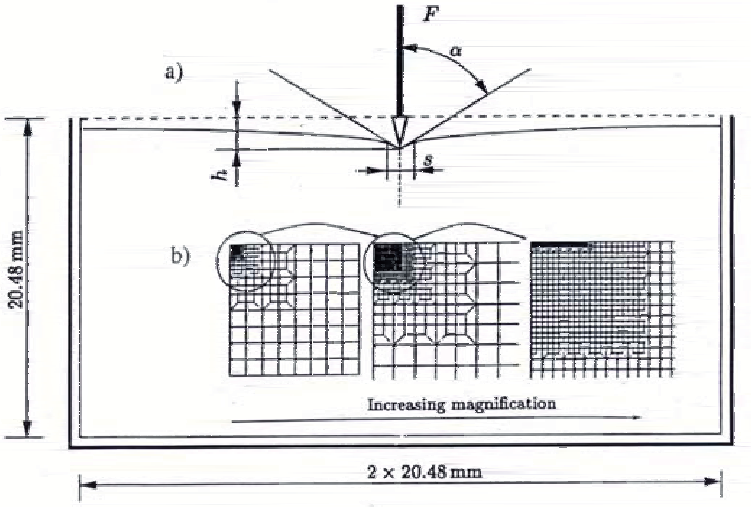


Figure 39. a) Indentation test, b) Increase of FE mesh density in the vicinity of indenter application

relationship $\bar{L}(\xi_1)$ is shown for the mean and $3\sigma_t = 3\sigma_N = 3\sigma_L$. It is visible that the distance between sigma bar curves is small in the region with a great amount of data.

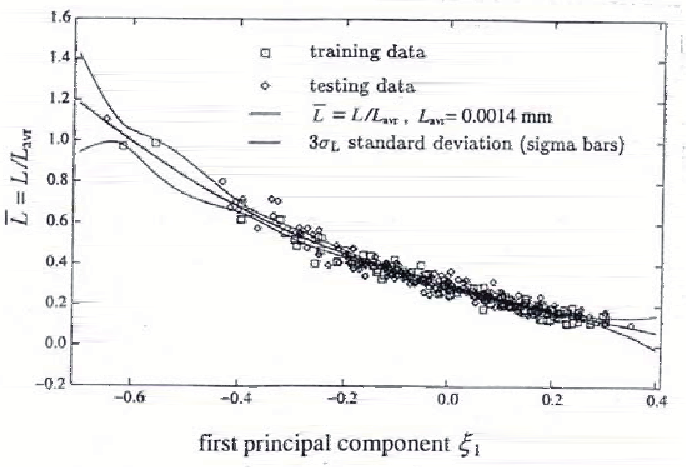


Figure 40. a) Mean and 3 sigma bar curves for relation $\bar{L}(\xi_1)$ computed by T-BNN

6.4 Selected problems from mechanics of concrete

In this Point two problems from mechanics of concrete are briefly discussed. The first problem concerns the design of High Performance Concrete (HPC) mixes from the point of view of concrete strength f_c' prediction. The other problem deals with estimation of the failure number of loading cycles applied to specimens made of ordinary concretes. In both presented problems the focus is put on applications of BNNs.

HCP strength prediction. The prediction of compressive strength f_c' of High Performance Concrete (HPC) by the standard NNs and BNNs is discussed in paper by Słoński (2007). 28 data sets composed of 346 mixes, collected by Kasperkiewicz et al. (1995), were used. The HPC data bases are related to six input variables and an output scalar variable:

$$\mathbf{x} = \{C, W, S, Su, FA, CA\}, \quad y = f_c', \tag{207}$$

where the input variables are amounts of the concrete mix ingredients in kg/m^3, corresponding to: C – cement, W – water, S – silica, Su – superplasiticizer, FA – fine aggregate, and CA – coarse aggregate. The output variable is the 28-day compressive strength f_c' [MPa].

The regression problem of HPC strength prediction was analysed in many papers, using also ANNs, see references in the PhD dissertation by Jakubek (2007). Now let us concentrate on the application of the Bayesian networks FLNN/MAP and T-BNN, see paper by Słoński (2009).

The network FLNN: 6-10-1 with bipolar sigmoid hidden neurons and linear output was used. The NETLAB Toolbox, see Nabney (2004) and MCMCStuff Toolbox, see Vehtari and Vanhatalo (2006) were used to both learn the networks FLNN, T-BNN and GP-BNN. The total number of patterns $P = 340$ was randomly split into the training and testing sets with $L = 226$ and $T = 114$ patterns, respectively. In the case of SNN the conjugate gradient method was applied and the computed weight vector $\mathbf{w}_{\mathrm{MAP}}$ gave values of errors and statistical parameters listed in Table 7.

Table 7. Errors of training and testing processes for neural prediction of HPC strength

ANN		Training (L)			Testing (T)		
Type and architecture	Learning method	*ARE* [%]	*RMS* [MPa]	r	*ARE* [%]	*RMS* [MPa]	r
FLNN: 6-10-1	Conj. Grad.	9.0	7.7	0.953	10.8	9.6	0.928
T-BNN: 6-10-1	HMC	6.6	5.8	0.974	9.3	8.0	0.951
GP-BNN	Conj. Grad.	6.6	5.6	0.976	11.3	10.2	0.918

For Bayesian neural network T-BNN, Gaussian noise model with constant variance was defined and hierarchical Gaussian prior was assumed. Learning and prediction were done using Gibbs sampling for hyperparameters and Hybrid Monte Carlo (HMC) method for weights. The MCMCStuff Toolbox software was applied, see Vehtari and Vanhatalo (2006). The main HMC parameters had the following values: length of chain was 100, step size was 0.5 and persistence parameter was 1.0. The burn-in stage contained 204 iterations and the actual sampling 800 iterations from which only 18 samples were used for HPC prediction.

The computations by T-BNN gave the values of errors and statistical parameters presented in Table 7.

For testing patterns also 1σ error bars were computed, corresponding to the estimated noise standard deviation ($\sigma_{BNN} = 7.3$ MPa vs. $\sigma_{SNN} = 7.7$ MPa). In Figure 41 the measured and computed values for both neural models are presented.

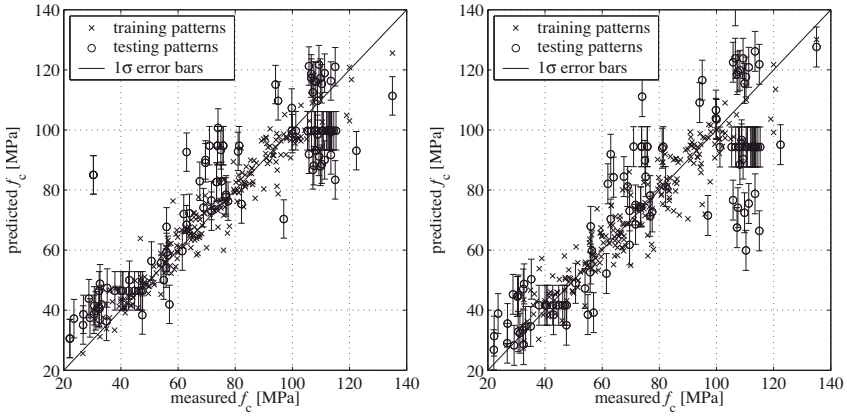


Figure 41. Predicted HPC compressive values vs. measured values using neural networks T-BNN (left) and FLNN/MAP (right)

On the base of discussed case study for HPC strength prediction some conclusions can be drawn. The Bayesian approach gave significantly better prediction of the mean value of HPC compressive strength comparing with predictions by the standard FLNNs. On the other hand, computations for Bayesian neural network T-BNN is much more labour-consuming than for the standard neural model.

Concrete fatigue failure prediction. The second example concerns prediction of concrete fatigue failure using the true Bayesian neural network T-BNN and Gaus-

sian Process model GP-BNN, see Słoński (2006, 2009). Concrete failure is defined as a number of loading cycles N which cause fatigue damage of a plain concrete specimen. This problem is based on a data set of 216 tests on concrete specimens, made in eight laboratories, see Furtak (1984). The patterns were randomly split into the training and testing sets with $L = 144$ and $T = 72$ patterns, respectively.

Following this paper, four input variables and a scalar output were adopted:

$$\mathbf{x} = \{f_c, \chi, R, f\}, \quad y = \log N, \tag{208}$$

where: $\chi = f_{cN}/f_c$ – ratio of compressive fatigue strength of concrete f_{cN} and strength f_c, $R = \sigma_{min}/\sigma_{max}$ – ratio of minimal and maximal strengths in compressive cycle of loading, f [Hz] – frequency of cyclic loading, N – number of load cycles associated with the fatigue failure.

Network FLNN: 4-7-1 was used of structure and neuron type selection as the network used in the problem discussed above. The GP model was defined using a squared exponential covariance function with the kernel components of the form similar to that defined in (183):

$$k(\mathbf{x}^n, \mathbf{x}^m) = \theta_0 \exp\Big(-\frac{1}{2}\sum_{i=1}^{4} \eta_i (x_i^n - x_i^m)^2\Big) + \theta_2. \tag{209}$$

Both Bayesian approaches, i.e Bayesian inference and Gaussian Process in the networks T-BNN and GP-BNN were applied. The computations were carried out by means of the NETLAB Toolbox, see Nabney (2004) and MCMCStuff Toolbox, see Vehtari and Vanhatalo (2006). In Table 8 the values of errors and statistical parameters are listed.

Table 8. Comparison of learning and testing errors and statistical parameters for standard network FLNN, Bayesian network T- BNN and Gaussian process based network GP-BNN

ANN		Training (L)			Testing (T)		
Type and architecture	Learning method	*ARE* [%]	*RMS* [MPa]	r	*ARE* [%]	*RMS* [MPa]	r
FLNN: 4-7-1	Conj. Grad.	12.4	0.64	0.894	14.0	0.72	0.857
T-BNN: 4-7-1	HMC	12.5	0.64	0.893	13.3	0.70	0.868
GP-BNN	Conj. Grad.	11.9	0.61	0.904	13.4	0.70	0.866

The true Bayesian neural network T-BNN and the network GB-BNN basing on the Gaussian process have very similar prediction capabilities, see Figure 42. It is worth emphasizing that learning and prediction for GP model is much easier to implement than for T-BNN.

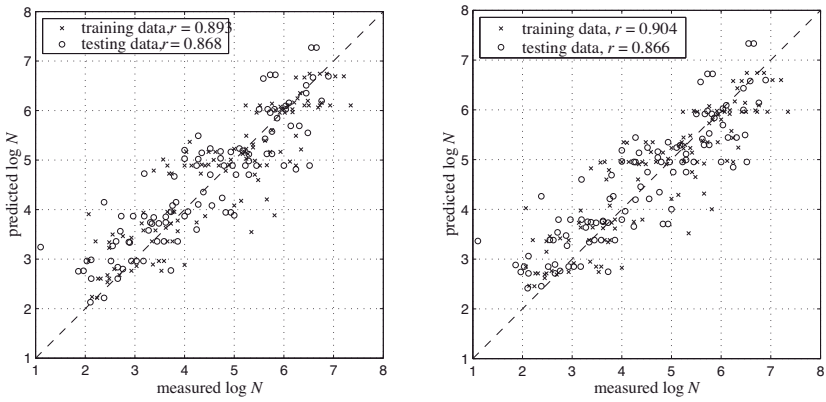


Figure 42. Predicted vs. measured fatigue failure of concrete for training and testing patterns for T-BNN (left) and GP-BNN (right)

Bibliography

Bailer-Jones, C., Sabin, T., MacKay, D. and Withers, P. (1997). Prediction of deformed and annealed microstructures using Bayesian neural networks and Gaussian processes. In *Proc. of the Australia-Pacific Forum on Intelligent Processing and Manufacturing of Materials*.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bishop, C. M. and Tipping, M. E. (2003). Bayesian regression and classification. In J. Suykens, S. B. C. M., G. Horvath and Vandewalle, J., editors, *Advances in Learning Theory: Methods, Models and Applications*, NATO Science Series III: Computer and Systems Sciences, pages 267–285. IOS Press.

Buntine, W. L. and Weigend, A. S. (1991). Bayesian back propagation. *Complex Systems*, 5(6):603–64.

Ciesielski, R., Kuźniar, K., Maciąg, E. and Tatara, T. (1992). Empirical formulae for fundamental natural periods of buildings with load bearing walls (in polish). *Archives of Civil Engineering*, 38:291–199.

Demuth, H. and Beale, M. (1998). *Neural Network Toolbox: For use with MATLAB: User's Guide, Version 3*. The Mathworks Inc.

Eurocode 8 (2003). Design of Structures for Earthquake Resistance.

Foresee, F. D. and Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian learning. In *IEEE International Conference on Neural Networks (IJCNN'97)*, volume III, pages III–1930–III–1935. IEEE.

Furtak, K. (1984). Strength of the concrete under multiple repeated loads, (in Polish). *Arch. of Civil Eng.*, 30.

Haykin, S. S. (1999). *Neural Networks: A Comprehensive Introduction*. Prentice Hall.

Haykin, S. S. (2001). *Kalman Filtering and Neural Networks*. John Wiley & Sons, Inc.

Jakubek, M. (2007). *Application of Neural Networks in Experimental Mechanics of Structures and Materials, (in Polish)*. Ph.D. thesis, Institute for Computational Civil Engineering, Cracow University of Technology.

Jang, J. S. R., Sun, C. T. and Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing*. Prentice Hall.

Kaczmarczyk, Ł. (2006). *Numerical Analysis of Multiscale Problems of Mechanics of Hetero-Homogeneous Continua (in Polish)*. Ph.D. thesis, Institute for Computational Civil Engineering, Cracow University of Technology.

Kaczmarczyk, Ł. and Waszczyszyn, Z. (2007). Identification of characteristic length of micro-structure for second order multiscale model by Bayesian neural networks. *Computer Assisted Mech. Eng. Sci.*, 14:183–196.

Kasperkiewicz, J., Racz, J. and A.Dubrawski (1995). HPC strength prediction using artificial neural network. *J. Comp. in Civ. Engrg.*, 9(4):1–6.

Korbicz, J., Obuchowicz, A. and Uciński, D. (1994). *Artificial Neural Networks: Foundations and Applications (in Polish)*. Akademicka Oficyna Wydawnicza.

Kouznetsova, V. (2002). *Computational Homogenization for the Multi-Scale Analysis of Multi-Phase Materials*. Ph.D. thesis, Technische Universiteit Eindhoven, The Netherlands.

Krok, A. (2006). Simulation of hysteresis loops for a superconductor using neural networks with Kalman filtering. *Computer Assisted Mech. Eng. Sci.*, 13:575–582.

Krok, A. (2007). *Analysis of Selected Problems of Mechanics of Structures and Materials by ANN and Kalman Filtering (in Polish)*. Ph.D. thesis, Institute for Computational Civil Engineering, Cracow University of Technology.

Krok, A. and Waszczyszyn, Z. (2007). Kalman filtering for neural prediction of response spectra from mining tremors. *Computers & Structures*, 85:1257–1263.

Kuźniar, K. (2003). BP Neural network computation of Response Spectra using a subpicture idea. In L., R. and J, K., editors, *Proc. Neural Networks and Soft Computing*, pages 754–759. T.U. of Częstochowa, Springer, Częstochowa/Zakopane.

Kuźniar, K. (2004). *Analysis of vibrations of medium height buildings subjected to mining tremors with application of neural networks (in Polish)*. Cracow University of Technology.

Kuźniar, K., Maciąg, E. and Waszczyszyn, Z. (2000). Computation of fundamental natural periods of vibrations of medium-hight buildings by neural networks. *Archives of Civil Engineering*, 46:515–523.

Kuźniar, K. and Waszczyszyn, Z. (2007). Neural networks for the simulation and identification of building subjected to paraseismic excitations. In Lagaros, N. D. and Tsompanakis, Y., editors, *Intelligent Computational Paradigms in Earthquake Engineering*. Idea Group Publishing.

Lampinen, J. and Vehtari, A. (2001). Bayesian approach for neural networks – review and case studies. *Neural Networks*, 14(3):7–24. (Invited article).

Lefik, M. (2005). *Application of Artificial Neural Networks in Mechanics and Engineering (in Polish)*. Łódź University of Technology.

Lefik, M. and Schrefler, B. (2002). One-dimensional model of cable-in-conduit superconductor under cyclic loading using artificial neural networks. *Fusion Engineering and Design*, 60:105–117.

Lou, K.-N. and Perez, R. (1996). A new system identification technique using Kalman filtering and multilayer neural networks. *Artificial Intelligence in Engineering*, 10:1–8.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

MacKay, D. J. C. (1998). Introduction to Gaussian processes. In Bishop, C. M., editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Springer.

MacKay, D. J. C. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Masters, T. (1993). *Practical Neural Network Recipes in C++*. Academic Press.

Nabney, I. T. (2004). *Netlab: Algorithms for Pattern Recognition*. Springer-Verlag, London.

Neal, R. M. (1992). Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Technical Report CRG-TR-92-1.

Neal, R. M. (2004). Software for Flexible Bayesian Modeling and Markov Chain Sampling. Technical report, University of Toronto.

Nijihuis, A., Noordman, N. and ten Kate, H. (1998). Mechanical and Electrical testing of an ITER CS1 Model Coil Conductor under Tramsferse Loading in a Cryogenic Press, Preliminary report. Technical report, University of Twente.

Pham, D. and Liu, X. (1995). *Neural Networks for Identification, Prediction and Control*. Springer.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts.

Rojas, R. (1996). *Neural Networks - A Systematic Introduction*. Springer.

Sato, T. and Sato, M. (1997). Structural identification using neural networks and kalman filtering. *JSCE*, 14:23–32.

Słoński, M. (2005). Prediction of concrete fatigue durability using Bayesian neural networks. *Computer Assisted Mech. Eng. Sci.*, 12:259–265.

Słoński, M. (2006). Bayesian regression approaches on example of concrete fatigue failure prediction. *Computer Assisted Mech. Eng. Sci.*, 13:655–668.

Słoński, M. (2007). HPC strength prediction using Bayesian neural networks. *Computer Assisted Mech. Eng. Sci.*, 14:345–352.

Słoński, M. (2009). A comparison between Bayesian neural networks and other machine learning methods for predicting properties of concrete. Proc. of *18th International Conference on Computer Methods in Mechanics*, CMM-2009, Zielona-Góra, Poland.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244.

Tipping, M. E. (2004). Bayesian Inference: An Introduction to Principles and Practice in Machine Learning. In O. Bousquet, U. v. L. and Rätsch, G., editors, *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 41–62. Springer.

Twomey, J. M. and Smith, A. E. (1997). Validation and verification. In Kartam, N., Flood, I. and Garrett, J. H., editors, *Neural Networks for Civil Engineers: Fundamentals and Applications*. ASCE, New York.

A. Vehtari and J. Vanhatalo. MCMC Methods for MLP and GP and Stuff (for Matlab) V2.1. A User Manual Laboratory of Computational Engineering, Helsinki University of Technology

Waszczyszyn, Z. (1999). *Neural Networks in the Analysis and Design of Structures*. CISM Courses and Lectures No. 404. Springer, Wien-New York.

Waszczyszyn, Z. (2006). Artificial neural networks in civil and structural engineering: Ten years of research in Poland. *Computer Assisted Mech. Eng. Sci.*, 13:489–512.

Waszczyszyn, Z. and Słoński, M. (2006). Bayesian neural networks for prediction of response spectra. *Foundations of Civil and Environmental Engineering*, 7:343–361.

Waszczyszyn, Z. and Ziemiański, L. (2005). Neural networks in the identification analysis of structural mechanics problems, Ch. 7. In Mróz, Z. and Stavroulakis, G., editors, *Parameter Identification of Materials and Structures*, CISM Lecture Notes No.469, pages 265–340. Springer, Wien - New York.

Zell, A., Mache, N., Sommer, T. and et. al. (1994). Stuttgart Neural Network Simulator. User manual, ver. 3.2. Technical report, University of Stuttgart, Germany.

Appendices

A1 Definitions of Errors for Discrete Sets of Data

The following error measures are used:

– *Mean-Square-Error* (MSE) and *Root-Mean-Square* error (RMS):

$$MSE \equiv \sigma^2_{\mathrm{ML}} \equiv \sigma^2_{\mathrm{LS}} \equiv \sigma^2_{\mathrm{MAP}} = \\ = \frac{1}{N}\sum_{n=1}^{N}\{t^n - y(\mathbf{x}^n;\mathbf{w})\}^2, \quad \text{(A1)}$$

$$RMS = \sqrt{MSE} \equiv \sigma_{\mathrm{ML}} \equiv \sigma_{\mathrm{LS}} \equiv \sigma_{\mathrm{MAP}} = \\ = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\{t^n - y(\mathbf{x}^n;\mathbf{w})\}^2}. \quad \text{(A2)}$$

– *Average Absolute Relative Error* (ARE)

$$ARE = \frac{1}{N}\sum_{n=1}^{N}\left|1 - y^n/t^n\right| \times 100\%. \quad \text{(A3)}$$

– *Coefficient of correlation* (linear regression)

$$r = \frac{\mathrm{cov}[\mathbf{x},\mathbf{y}]}{\sigma_x \sigma_y} = \frac{\sum_{n=1}^{N}(x^n - \bar{x})\sum_{n=1}^{N}(y^n - \bar{y})}{\sqrt{\sum_{n=1}^{N}(x^n - \bar{x})^2 \sum_{n=1}^{N}(y^n - \bar{y})^2}}. \quad \text{(A4)}$$

where:

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x^n, \quad \bar{y} = \frac{1}{N}\sum_{n=1}^{N} y^n - \text{means of vectors } \mathbf{x} \text{ and } \mathbf{y}. \quad \text{(A5)}$$

A2 Acronyms

ANN	Artificial Neural Network
BNN	Bayesian NN
BP	Back Propagation
DEKF	Decoupled Extended Kalman Filter
FLNN	Feed-forward Layered NN
GEKF	General Extended Kalman Filter
i.i.d.	independent and identically distributed events
Mal	Marginal likelihood
MAP	Maximum APosterior
ML	Maximum Likelihood
MP	Most Probable
MSE	Mean Square Error
PCA	Principal Component Analysis
pd	probability distribution
RBF	Radial Basis Function
RFSN	Radial Basis Function Neural network
RMS	Root Mean Square error
S-BNN	Simple BNN
SNN	Standard (deterministic) Neural Network
SNN/MAP	SNN with use of MAP
SNN/ML	SNN with use of ML
T-BNN	True BNN

CHAPTER 6

Neural Networks: Some Successful Applications in Computational Mechanics

Manolis Papadrakakis, Nikos D. Lagaros and Michalis Fragiadakis

Institute of Structural Analysis & Seismic Research,
School of Civil Engineering,
National Technical University Zografou Campus, Athens 15780, Greece

Abstract This article presents recent applications of neural computations in the field of stochastic finite element analysis of structures and earthquake engineering. The incorporation of Neural Networks (NN) in this type of problems is crucial since it leads to substantial reduction of the excessive computational cost. Earthquake-resistant design of structures using Probabilistic Safety Analysis (PSA) is an emerging field in structural engineering. The efficiency of soft computing methodologies is investigated when incorporated into the solution of computationally intensive earthquake engineering problems considering uncertainties.

1 Introduction

Over the last ten years artificial intelligence techniques like Neural Networks (NN) have emerged as a powerful tool that could be used to replace time consuming procedures in many scientific or engineering applications. The fields where NN have been successfully applied are: (i) pattern recognition, (ii) regression (function approximation/fitting) and (iii) optimization. In the past the application of NN was mostly used for predicting the behavior of structural systems in the context of structural optimal design (McCorkle et. al 2003, Papadrakakis and Lagaros 2002], structural damage assessment (Zacharias et. al 2004), the evaluation of buckling loads of imperfect structures (Waszczyszyn et al. 2002) or structural reliability analysis (Hurtado and Alvarez 2002, Nie and Ellingwood 2004,Papadrakakis et al. 1996). This study presents recent developments in the applications of NN in the field of stochastic finite element and probabilistic analysis of structures.

Many sources of uncertainty (material, geometry, loads, etc) are inherent in structural systems. Probabilistic analysis of structures leads to safety measures that a design engineer has to take into account due to the aforementioned uncertainties. Probabilistic analysis problems, especially when seismic loading

is considered, are highly computationally intensive tasks since, in order to obtain the structural behaviour, a large number of dynamic analyses (e.g modal response spectrum analysis, or nonlinear timehistory analysis) are required. In this work two metamodel-based applications are considered in order to reduce the excessive computational cost. The efficiency of a trained NN is demonstrated, where a network is used to predict maximum interstorey drift values due to different sets of random variables. As soon as the maximum interstorey drift is known, the limit-state probabilities are calculated by means of Monte Carlo Simulation (MCS). In the first application the probability of exceedance of a limit-state is obtained when the Multi-modal Response Spectrum analysis is adopted (Tsompanakis et al. 2008). In the second application fragility analysis of a ten-storey moment resisting steel frame is evaluated where limit-state fragilities are determined by means of nonlinear time history analysis (Lagaros and Fragiadakis 2007).

The field of structural reliability has been developed significantly during the last twenty years and has been documented in an increasing number of publications (Schuëller 2005). In this work the probabilistic safety analysis of framed structures under seismic loading conditions is investigated based on the methodology proposed by Lagaros and Fragiadakis (2007). Both randomness of ground motion excitation (that influence the seismic demand level) and material properties (that affect the structural capacity) are taken into consideration. Additionally, a computationally efficient procedure, proposed in a previous work by Lagaros *et al.* (2005), for the simulation of homogeneous non-Gaussian stochastic fields with prescribed target marginal distribution and spectral density function is implemented.

The assessment of the bearing capacity of framed structures, in terms of maximum interstorey drift, is determined via non-linear time history analysis. Probabilistic Safety Analysis (PSA) using the Monte-Carlo Simulation (MCS) method and non-linear time history analysis results in a highly computationally intensive problem. In order to reduce the computational cost, NN are employed. For the training of the NN a number of Intensity Measures (IMs) are used in order to accurately predict the maximum interstorey drift values. The IMs adopted in the present study can be classified either as seismic record dependent, or as both structure and record dependent. Via the presented PSA procedure fragility curves are obtained for different hazard levels. In addition the probability of structure's failure is derived as a limit state function of seismic intensity.

2 Multi-layer Perceptrons

A multi-layer perceptron is a feed-forward neural network consisting of a number of units (neurons) linked together. Training attempts to create a desired

relation in an input/output set of learning patterns. A learning algorithm tries to determine the weight parameters, in order to achieve the right response for each input vector applied to the network. The numerical minimization algorithms used for the training generate a sequence of weight matrices through an iterative procedure. To apply an algorithmic operator A a starting weight matrix $w^{(0)}$ is needed, while the iteration formula can be written as follows

$$w^{(t+1)}=A(w^{(t)})=w^{(t)}+\Delta w^{(t)} \tag{1}$$

All numerical methods applied for the NN training are based on the above formula. The changing part of the algorithm $\Delta w^{(t)}$ is further decomposed into two parts as

$$\Delta w^{(t)}=a_t d^{(t)} \tag{2}$$

where $d^{(t)}$ is a desired search direction of the move and a_t the step size in that direction.

The training methods can be divided into two categories. Algorithms that use global knowledge of the state of the entire network, such as the direction of the overall weight update vector, which are referred to as *global* techniques. In contrast, local adaptation strategies are based on weight specific information only, such as the temporal behaviour of the partial derivative of this weight. The local approach is more closely related to the NN concept of distributed processing in which computations can be made independent to each other. Furthermore, it appears that for many applications local strategies achieve faster and reliable prediction than global techniques despite the fact that they use less information (Schiffmann et al. 1993).

2.1 Global Adaptive Techniques

The algorithms most frequently used in the NN training are the steepest descent, the conjugate gradient and the Newton's methods with the following direction vectors:

Steepest descent method: $d^{(t)} = -\nabla E(w^{(t)})$

Conjugate gradient method: $d^{(t)} = -\nabla E(w^{(t)}) + \beta_{t-1} d^{(t-1)}$ where β_{t-1} is defined as:

$$\beta_{t-1} = \nabla E_t \cdot \nabla E_t / \nabla E_{t-1} \cdot \nabla E_{t-1} \text{ Fletcher-Reeves}$$

Newton's method: $d^{(t)} = -\left[H(w^{(t)})\right]^{-1} \nabla E(w^{(t)})$.

The convergence properties of the optimization algorithms for differentiable functions depend on the properties of the first and/or second derivatives of the function to be optimized. When optimization algorithms converge slowly for NN problems, this suggests that the corresponding derivative matrices are numerically ill-conditioned. It has been shown that these algorithms converge

slowly when rank-deficiencies appear in the Jacobian matrix of a NN, making the problem numerically ill-conditioned (Lagaros and Papadrakakis 2004).

2.2 Local Adaptive Techniques

To improve the performance of weight updating, two approaches have been proposed, namely Quickprop (Fahlman 1988) and Rprop (Riedmiller and Braun 1993).

The Quickprop method

This method is based on a heuristic learning algorithm for a multi-layer perceptron, developed by Fahlman (1988), which is partially based on the Newton's method. Quickprop is one of most frequently used adaptive learning paradigms. The weight updates are based on estimates of the position of the minimum for each weight, obtained by solving the following equation for the two following partial derivatives

$$\frac{\partial E_{t-1}}{\partial w_{ij}} \text{ and } \frac{\partial E_t}{\partial w_{ij}} \tag{3}$$

and the weight update is implemented as follows

$$\Delta w_{ij}^{(t)} = \frac{\dfrac{\partial E_t}{\partial w_{ij}}}{\dfrac{\partial E_{t-1}}{\partial w_{ij}} - \dfrac{\partial E_t}{\partial w_{ij}}} \Delta w_{ij}^{(t-1)} \tag{4}$$

The learning time can be remarkably improved compared to the global adaptive techniques.

The Rprop method

Another heuristic learning algorithm with locally adaptive learning rates based on an adaptive version of the Manhattan-learning rule and developed by Riedmiller and Braun (1993) is the **R**esilient back**prop**agation abbreviated as Rprop. The weight updates can be written

$$\Delta w_{ij}^{(t)} = -\eta_{ij}^{(t)} \operatorname{sgn}\left(\frac{\partial E_t}{\partial w_{ij}}\right) \tag{5}$$

where

$$\eta_{ij}^{(t)} = \begin{cases} \min(\alpha \cdot \eta_{ij}^{(t-1)}, \eta_{max}), & \text{if } \frac{\partial E_t}{\partial w_{ij}} \cdot \frac{\partial E_{t-1}}{\partial w_{ij}} > 0 \\ \max(b \cdot \eta_{ij}^{(t-1)}, \eta_{min}), & \text{if } \frac{\partial E_t}{\partial w_{ij}} \cdot \frac{\partial E_{t-1}}{\partial w_{ij}} < 0 \\ \eta_{ij}^{(t-1)}, & \text{otherwise} \end{cases} \tag{6}$$

where α=1.2, b= 0.5, η_{max}=50 and η_{min}=0.1 (Riedmiller, 1994). The learning rates are bounded by upper and lower limits in order to avoid oscillations and arithmetic underflow. It is interesting to note that, in contrast to other algorithms, Rprop employs information about the sign and not the magnitude of the gradient components.

3 Fragility Analysis using Monte Carlo Simulation

Extreme earthquake events may produce extensive damage to structural systems despite their low probabilities of occurrence. It is therefore essential to establish a reliable procedure for assessing the seismic risk of real-world structural systems. Probabilistic safety analysis provides a rational framework for taking into account the various sources of uncertainty that may influence structural performance under seismic loading conditions. The core of PSA is seismic fragility analysis, which provides a measure of the safety margin of a structural system for different limit states.

In this section the probabilistic safety analysis of framed structures under seismic loading conditions is investigated. Randomness of ground motion excitation (that influences seismic demand) and of material properties (that affect structural capacity) are taken into consideration using Monte Carlo Simulation. The capacity assessment of steel frames is determined using nonlinear timehistory analysis. The probabilistic safety analysis using Monte-Carlo Simulation and nonlinear time history analysis results in a computationally intensive problem. In order to reduce the excessive computational cost, techniques based on NN are implemented. For the training of the NN a number of IMs are derived from each earthquake record, for the prediction of the level of damage, which is measured by means of maximum interstorey drift values θ_{max}.

The seismic fragility of a structure $F_R(x)$ is defined as its limit-state probability, conditioned on a specific peak ground acceleration, spectral velocity, or other control variable consistent with the specification of seismic hazard

$$F_R(x) = P[LS_i / PGA = x] \tag{7}$$

where LS_i represents the corresponding ith limit state and the peak ground PGA is the control variable. If the annual probabilities of exceedance $P[PGA=x]$ of specific levels of earthquake motion are known, then the mean annual frequency of exceedance of the ith limit state is calculated as follows:

$$P[LS_i] = \sum_x F_R(x) P[PGA = x] \tag{8}$$

Eq. (8) can be used for taking decisions about, for example, the adequacy of a design or the need to retrofit a structure. In the present study the aim is to evaluate the fragility F_R(x). Once the fragility is calculated the extension to Eq. (8) is straightforward.

Often F_R(x) is modelled with a lognormal probability distribution, which leads to an analytic calculation. In the present study Monte Carlo Simulation (MCS) with improved Latin Hypercube Sampling (iLHS) for the reduction of the sampling size, is adopted for the numerical calculation of F_R(x). Numerical calculation of Eq. (7) provides a more reliable estimate of the limit state probability, since it is not necessary to assume that seismic data follow a lognormal distribution. However, in order to calculate the limit state probability, a large number of nonlinear dynamic analyses are required for each hazard level, especially when the evaluation of extremely small probabilities is needed.

The methodology requires that MCS has to be performed at each hazard level. Earthquake records are selected randomly and scaled to a common intensity level that corresponds to the hazard level examined. Scaling is performed using the first mode spectral acceleration of the 5% damped spectrum ($Sa(T_1,5\%)$). Therefore, all records are scaled in order to represent the same ground motion intensity in terms of $Sa(T_1,5\%)$. Earthquake loading is considered as two separate sources of uncertainty, ground motion intensity and the details of ground motion. The first uncertainty refers to the general severity of shaking at a site, which may be measured in terms of any IM such as PGA, $Sa(T_1,5\%)$, Arias intensity, etc. The second source refers to the fact that, although different acceleration time histories can have their amplitudes scaled to a common intensity, there is still uncertainty in the performance, since IMs are imperfect indicators of the structural response. The first source is considered by scaling all records to the same intensity level at each limit state. The second source is treated by selecting natural records as random variables from a relatively large suite of scenario based records. The concept of considering separately seismic intensity and the details of ground is the backbone of the Incremental Dynamic Analysis (IDA) method (Vamvatsikso and Cornell 2002), while Porter *et al.* (2002) have also introduced intensity and different records as two separate uncertain parameters in order to evaluate the sensitivity of

structural response to different uncertainties.

The random parameters considered in this study are the material properties and more specifically the modulus of elasticity E and the yield stress f_y, as well as the details of ground motion where a suite of scenario based earthquake records is used. The material properties are assumed to follow the normal distribution while the uniform distribution is assumed for the records in order to select them randomly from a relatively large bin of natural records. The first two variables are sampled by means of the iLHS technique in order to increase the efficiency of the sampling process.

In reliability analysis the MCS method is often employed when the analytical solution is not attainable and the failure domain can not be expressed or approximated by an analytical form. This is mainly the case in problems of complex nature with a large number of basic variables where all other reliability analysis methods are not applicable. Expressing the limit state function as $\mathbf{G}(\mathbf{x})<0$, where $\mathbf{x}=(x_1,x_2,...,x_M)^T$ is the vector of the random variables, the probability of exceedance can be obtained as

$$P_{LS} = \int_{G(x)\geq 0} f_x(x)dx \tag{9}$$

where $f_x(\mathbf{x})$ denotes the joint probability of failure for all random variables. Since MCS is based on the theory of large numbers (N_∞) an unbiased estimator of the probability of failure is given by

$$P_{LS} = \frac{1}{N_\infty}\sum_{j=1}^{N_\infty} I(x_j) \tag{10}$$

where $I(\mathbf{x}_j)$ is a Boolean vector indicating failure and non-failure simulations. In order to estimate P_{LS} an adequate number of N_{sim} independent random samples is produced using a specific probability density function for the vector $\mathbf{x}$. The value of the failure function is computed for each random sample x_j and the Monte Carlo estimation of P_{LS} is given in terms of the sample mean by

$$P_{LS} \cong \frac{N_H}{N_{sim}} \tag{11}$$

where N_H is the number of failure simulations, where the maximum interstorey drift value exceeds a threshold drift for the limit state examined. In order to calculate Eq. (11) N_{sim} nonlinear time history analyses have to be performed at each hazard level. Clearly the computational cost of performing so many nonlinear dynamic analyses, even when an efficient sampling reduction technique (such as iLHS) is used, is prohibitive. In order to reduce the computational cost, properly trained NN are implemented.

As already mentioned back-propagation NN are used in order to reduce the number or earthquake simulations required for the calculation of the probability of Eq. (11). The principal advantage of a properly trained NN is that it requires a trivial computational effort to produce an acceptable approximate solution. Such approximations appear to be valuable in situations where actual response computations are CPU intensive and quick estimations are required. Neural networks have been applied in the past by Papadrakakis *et al.* (1996) in order to calculate the probability of failure for steel moment frames using inelastic static analysis. In recent studies NN have been adopted for the reliability analysis of structures by Nie and Ellingwood (2004) and Hurtado (2001). However, in the present study the NN are implemented in order to predict the maximum seismic response with natural earthquake records replacing the time consuming nonlinear time history analysis. The NN are trained in order to predict the maximum interstorey drift $\theta_{\max}$ for different earthquake records which are identified by NN using a set of IMs.

The term Intensity Measure is used to denote a number of common ground motion parameters which represent the amplitude, the frequency content, the duration or any other ground motion parameter. A number of different IMs has been presented in the literature (Kramer 1996), while various attempts to relate an IM with a damage measure such as maximum interstorey drift values have been made (Shome and Cornell, 1999). The IMs adopted can be classified as record dependent only or as both structure and record dependent. The complete list of the IMs used in this study is given in Table 1.

Table 1. Intensity measures

No	Intensity Measure
1	PGA (g)
2	PGV (m)
3	PGD (m)
4	V/A (sec)
5	Arias intensity (m/sec)
6	Significant duration (5 to 95 % of Arias) (sec)
7	RMS acceleration (g)
8	Characteristic Intensity
9	CAV
10	Spectral Intensity
11	Total Duration (sec)
12	SA(T_1) (g)
13	SV(T_1) (cm)
14	SaC, c=2 (g)
15	SaC, c=3 (g)

It can be seen that the IMs selected, vary from widely used ground motion parameters such as peak ground acceleration (*PGA*) to more sophisticated measures such as *SaC*. The definitions and further discussion on the first

thirteen measures of Table 1 is given by Kramer (1996). The last two IMs refer to the measure are defined as

$$SaC = Sa(T_1)\sqrt{\frac{Sa(c \cdot T_1)}{Sa(T_1)}} \tag{12}$$

The parameter c takes the value 2 and 3 for the 14th and the 15th parameter of Table 1, respectively. These IMs were introduced in order to assist the NN to capture the effects of inelasticity by considering the elastic spectrum at an "effective" period longer than T_1, thus reflecting the reduction in stiffness.

For each hazard level separate training of the NN is performed by means of the above IMs. The training process is based on the fact that the trained NN will assign small weights to the IMs which have poor correlation with the damage measure selected. Instead of using the whole set it was examined the suitability of using only some of the IMs of Table 1. The parametric study was performed for various intensity levels since the performance of an IM depends also on the level of nonlinearity that the structure has undergone. The ten combinations of IMs, shown in Table 2, were compared.

Table 2. Intensity measures combinations

ID	IM combinations
A	1
B	1,2
C	1,2,3
D	1,2,3,5
E	1,2,3,5,9
F	1,2,3,5,9,10
G	1,2,3,5,9,10,12
H	1,2,3,5,9,10,12,14
I	1,2,3,5,9,10,12,14,15
J	ALL

The performance of each combination is shown in Table 3. The efficiency of the NN is evaluated for the ten-storey steel moment resisting frame described in one of the next sections. For this parametric study the material random variables were considered with their mean values. From Table 3 it is clear that the use of record dependent only measures, such as *PGA*, lead to increased error values, while more refined measures help to reduce the error considerably. The use of the complete set of IMs in Table 1 is preferred since it performed equally well for all four hazard levels examined in the parametric study.

4 NN-based Seismic Fragility Analysis

A suite of 95 scenario-based natural records were used in this study. In order to obtain the fragility curves, sixteen hazard levels expressed in PGA terms

ranging from 0.05g to 1.25g were used. For each hazard level, risk assessment is performed and five limit state fragilities are calculated. Each limit state is defined by means of a corresponding maximum interstorey drift θ_{max} value. In the present study the five limit states considered range from serviceability, to life safety and finally to the onset of collapse. The corresponding θ_{max} threshold values range from 0.2 to 6 percent.

Table 3. Prediction errors for the maximum interstorey drift θ_{max}

	IM Combination									
	A	B	C	D	E	F	G	H	I	ALL
	PGA = 0.05g									
MAX	49.7	39.6	19.6	32.6	15.9	14.8	4.9	27.0	23.6	9.0
MIN	4.4	0.2	0.9	0.3	0.3	0.6	1.0	1.8	0.6	0.3
AVERAGE	26.2	11.0	7.8	7.6	6.6	6.3	2.9	8.5	7.9	4.4
	PGA = 0.27g									
MAX	32.6	28.2	21.4	26.6	46.7	23.6	9.5	24.5	35.9	9.6
MIN	0.9	0.1	1.1	0.5	1.4	0.1	0.6	0.7	0.3	1.5
AVERAGE	16.9	15.7	11.1	13.3	13.4	9.3	5.0	10.3	9.9	5.0
	PGA = 0.56g									
MAX	62.0	67.2	28.8	42.2	35.4	28.0	33.2	29.3	16.4	9.2
MIN	6.7	3.2	0.2	0.6	0.1	0.2	0.7	1.3	2.2	0.8
AVERAGE	22.8	26.8	10.7	13.1	18.2	14.3	11.1	10.5	7.5	4.8
	PGA = 0.90g									
MAX	72.1	45.2	51.0	23.3	13.3	16.9	12.0	8.7	12.5	9.2
MIN	3.0	6.1	1.4	0.5	1.2	0.9	0.5	0.6	0.2	0.9
AVERAGE	35.9	19.8	18.4	6.8	4.9	8.7	3.8	3.9	4.4	3.8

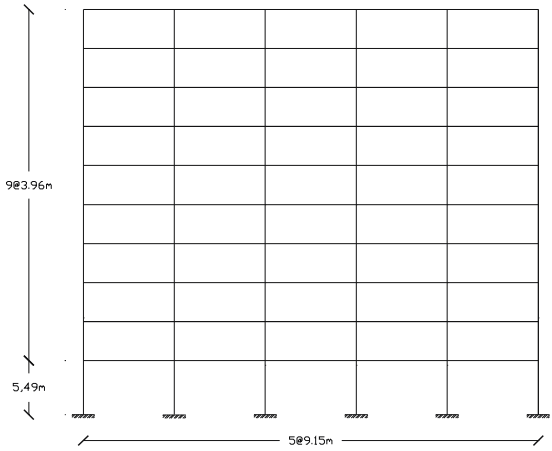


Figure 1. Ten-storey steel moment frame

The test example considered to demonstrate the efficiency of the procedure is the five-bay, ten-storey moment resisting plane frame of Figure 1. The mean values of the modulus of elasticity is equal to 210 GPa and the yield stress is f_y=235 MPa. The coefficients of variation for E and f_y are considered as 5% and 10%, respectively. The constitutive law is bilinear with a strain hardening ratio

of 0.01, while the frame is assumed to have rigid connections and fixed supports. The permanent load is equal to 5kN/m^2 and the live load is taken as Q = 2kN/m^2. The gravity loads are contributed from an effective area of 5m. All analyses were performed using a force-based fiber beam-column element (Papaioannou et al. 2005) that allows the use of a single element per member, while the same material properties are used for all the members of the frame.

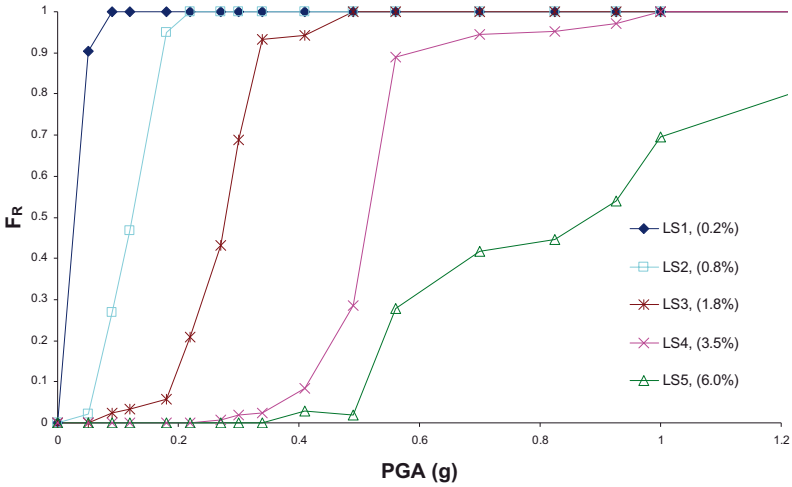


Figure 2. Fragility curves

For training the NN both training and testing sets have to be selected for each hazard level. The selection of the sets is based on the requirement that the full range of possible results has to be taken into account in the training step. Therefore, training/testing triads of the material properties and the records are randomly generated using the Latin Hypercube sampling. In the case of earthquake records the selection has to take into account that the scaling factor should be between 0.2 and 5. This restriction is applied because large scaling factors are likely to produce unrealistic earthquake ground motions. Furthermore, the records selected for generating the training set have to cover the whole range of structural damage for the hazard level in consideration. Thus, nonlinear time history analyses were performed, for mean E and f_y values, where the θ_{max} values of each record that satisfy the previous requirement were determined for each hazard level. In total 30 records are selected for generating the training set of each hazard level taking into account that the selection has to cover the whole range of θ_{max} values. Therefore, training sets with 90 triads of E, f_y and record number, all sampled as discussed above, are generated. Finally, a testing sample of 10 triads is also selected in a similar way in order to test the performance of the NN.

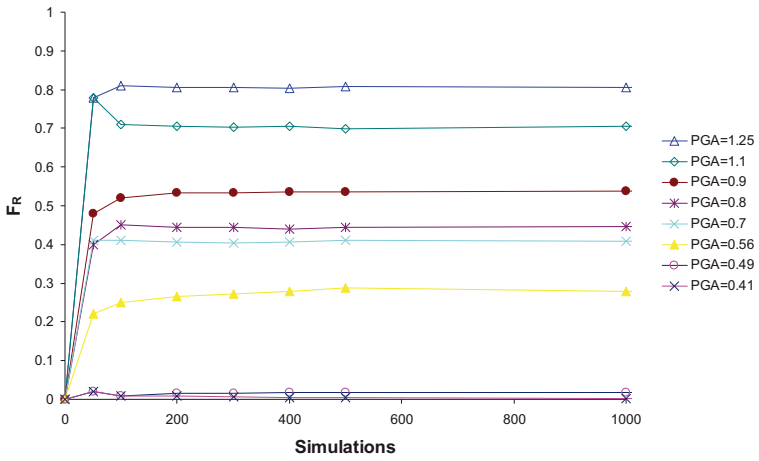


Figure 3. Number of NN-Simulations required ($\theta_{max} \geq 6.0\%$)

The fragility curves obtained for the five limit states considered are shown in Figure 2. Figure 3 shows the number of MCS simulations required for the fragility curve of a one limit state, in particular the Near Collapse limit state ($\theta_{max} \geq 6.0\%$). It can be seen that depending on the calculated probability of exceedance the number simulations required for a single point of the fragility curve, ranges from 50 to 1000. The validity of the prediction obtained with the NN is shown in Figure 4. The maximum interstorey drift values predicted for the 10 components of the testing set compared to the values obtained with nonlinear time-history analysis are shown in Figure 2 for four hazard levels.

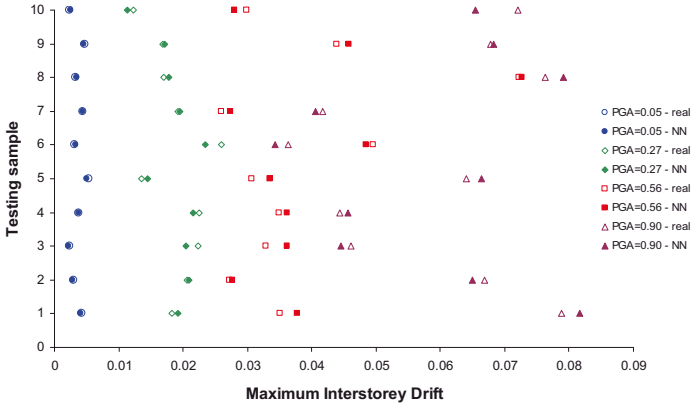


Figure 4. Prediction of θ_{max} for the testing sample

5 Metamodel Assisted Methodology for Validating the EC8 Approach

Extreme earthquake events may produce extensive damage to structural systems. It is therefore essential to establish a reliable procedure for assessing the seismic risk of real-world structural systems. The reliability of a steel frame designed to EC8 using the modal response analysis as suggested by Eurocode 8 is performed based on the work of Tsompanakis *et al.* (2008). The probability that the life-safety is exceeded is determined, since this is the limit-state that usually controls the design process.

5.1 Metamodel Assisted Methodology

In the present implementation the main objective is to investigate the ability of the NN to predict the structural performance in terms of maximum interstorey drift. The selection of appropriate I/O training data is an important part of the NN training process. Although the number of training patterns may not be the only concern, the distribution of samples is of greater importance. In the present study the sample space for each random variable is divided into equally spaced distances in order to select suitable training pairs. Having chosen the NN architecture and tested the performance of the trained network, predictions of the probability of exceedance of the design limit-state can be made quickly. The results are then processed by means of MCS to calculate the probability of exceedance p_{exceed} of the design limit-state using Eq. (11).

Table 4. Input and output variables for the two levels of approximation.

Test case	1st level of approximation (NN1)		2nd level of approximation (NN2)	
	Inputs	Outputs	Inputs	Outputs
(a)	E	T_i, i=1,…,8	$R_{dx}(T_i)$, $R_{dy}(T_i)$, i=1,…,8	Max drift θ_{max}
(b)	b_i, h_i, i=1,…,5	T_i, i=1,…,8	$R_{dx}(T_i)$, $R_{dy}(T_i)$, i=1,…,8	Max drift θ_{max}
(c)	E, b_i, h_i, i=1,…,5	T_i, i=1,…,8	$R_{dx}(T_i)$, $R_{dy}(T_i)$, i=1,…,8	Max drift θ_{max}

The modulus of elasticity, the dimensions b and h of the I-shape cross-section and the seismic loading have been considered as random variables. Three alternative test cases are considered depending on the random variables considered: (a) the modulus of elasticity and the earthquake loading are considered to be random variables, (b) the dimensions b and h of the I-shape cross section and earthquake loading are taken as random variables and (c) all three groups of random variables are considered together. For the implementation of the NN-based metamodel in all three test cases a two level

approximation is employed using two different NN. The first NN predicts the eigenperiod values of the significant modes. The inputs of the NN are the random variables while the outputs are the eigenperiod values. The second NN is used to predict the maximum interstorey drift, which is used to determine whether a limit-state has been violated. Therefore, the spectral acceleration values of both X and Y directions are the input values of the NN while the maximum interstorey drift is the output. The input and output variables for the two levels of approximation are shown in Table 4.

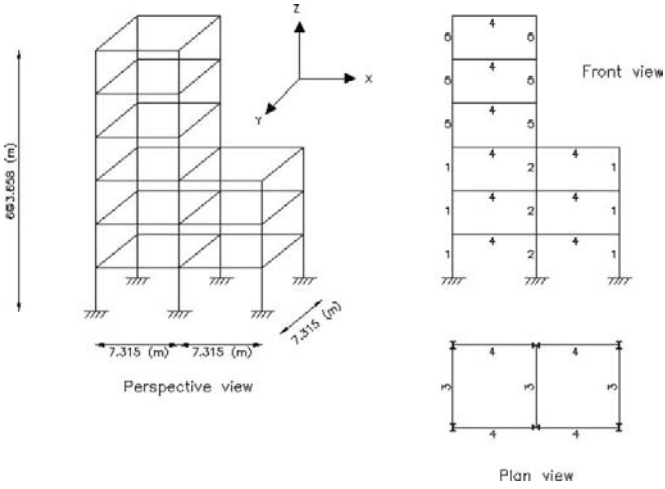


Figure 5. The six-storey space frame

5.2 Seismic Probabilistic Analysis

The six storey space frame, shown in Figure 5**,** has been considered to assess the proposed metamodel-assisted structural probabilistic analysis methodology. The space frame consists of 63 members that are divided into five groups having the following cross sections: (1) IPB 650, (2) IPB 650, (3) IPE 450, (4) IPE 400 and (5) IPB 450. The structure is loaded with a permanent action of G = 3 kN/m^2 and a live load of Q = 5 kN/m^2. In order to take into account that structures may deform inelastically under earthquake loading, the seismic actions are reduced using a behavior factor q = 4.0 (Eurocode 8, 1994).

The most common way of defining the seismic loading is by means of a regional design code response spectrum. However if higher precision is required the use of spectra derived from natural earthquake records is more appropriate. Therefore, a set of nineteen natural accelerograms, shown in Table 5, is used.

Table 5. List of the natural records

Earthquake	Station	Distance	Site
Tabas 16 Sept. 1978	Dayhook	14	rock
	Tabas	1.1	rock
Cape Mendocino 25 April 1992	Cape Mendocino	6.9	rock
	Petrolia	8.1	soil
Chi-Chi 20 Sept. 1999	TCU052	1.4	soil
	TCU065	5.0	soil
	TCU067	2.4	soil
	TCU068	0.2	soil
	TCU071	2.9	soil
	TCU072	5.9	soil
	TCU074	12.2	soil
	TCU075	5.6	soil
	TCU076	5.1	soil
	TCU078	6.9	soil
	TCU079	9.3	soil
	TCU089	7.0	rock
	TCU101	4.9	soil
	TCU102	3.8	soil
	TCU129	3.9	soil

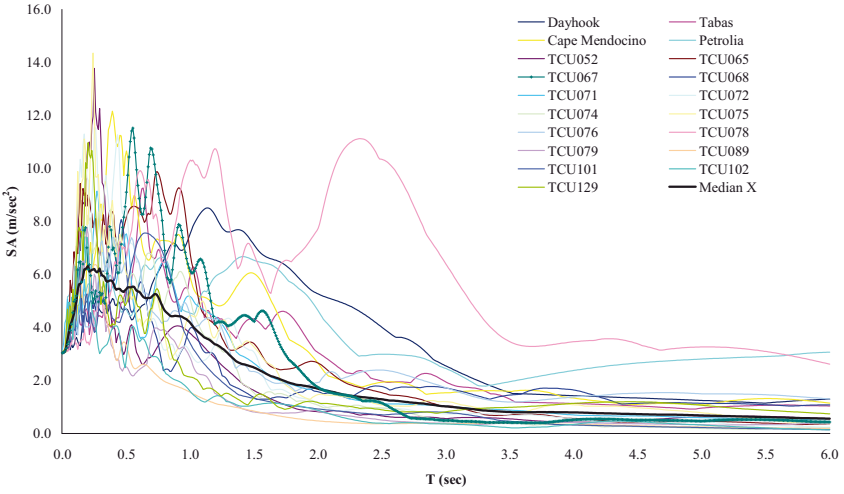


Figure 6. Natural record response spectra and their median spectrum (Longitudinal components)

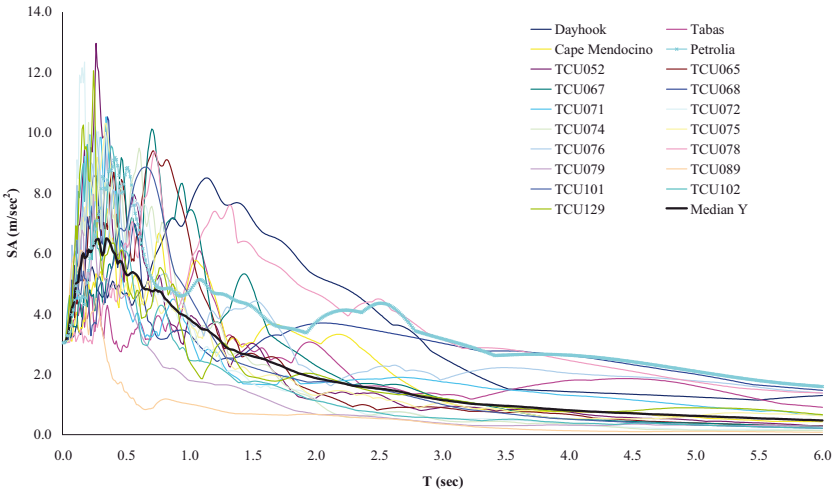


Figure 7. Natural record response spectra and their median spectrum (Transverse components)

The records are scaled, to the same peak ground acceleration of 0.32g in order to ensure compatibility. Two are from the 1992 Cape Mendocino earthquake, two are from the 1978 Tabas, Iran earthquake and fifteen are from the 1999 Chi-chi, Taiwan earthquake. The response spectra for each scaled record, in X and T directions, are shown in Figures 6 and 7, respectively. In Table 6 the probability density functions, mean values and standard deviations for all random variables are listed.

Table 6. Characteristics of the random variables.

Random variable	Probability density function	Mean value	Standard deviation
E	N	210	10 (%)
b	N	b*	2 (%)
h	N	h*	2 (%)
Seismic load	Log-N	$\hat{x}$, (Eq. 13)	δ, (Eq. 14)

* Dimensions from the IPE and HEB databases

Each record corresponds to different earthquake magnitudes and soil properties corresponding to different earthquake events. The chosen set is kept in rational size in order to achieve increased efficiency of NN with minimum data and covers, as much as possible, a wide range of excitations in order to provide NN with more "widespread" information as it is well known that NN

cannot extrapolate efficiently any given information. Of course, as in any kind of NN application, the selection of approximation data, i.e. the set of records (both the ones for training and the other for NN approximations) in the present application, affects significantly the performance of NN.

Assuming that seismic loading data are distributed lognormally the median spectrum $\hat{x}$ and the standard deviation δ are calculated as follows:

$$\hat{x} = \exp\left[\frac{\sum_{i=1}^{n} \ln(R_{d,i}(T))}{n}\right] \tag{13}$$

$$\delta = \left[\frac{\sum_{i=1}^{n} \left(\ln(R_{d,i}(T)) - \ln(\hat{x})\right)^2}{n-1}\right]^{1/2} \tag{14}$$

where $R_{d,i}(T)$ is the response spectrum of the i^{th} record for period value equal to T. The median spectra for both directions are shown in Figures 6 and 7.

In the first test case one hundred (100) training/testing patterns of the modulus of elasticity are selected in order to train the first NN and one hundred (100) training/testing patterns of the spectral values are selected to train the second NN. Ten out of them are selected to test the efficiency of the trained network. In the second test case the training-testing set was composed by one hundred fifty (150) pairs while in the third test case the set was composed by two hundred (200) pairs while in both cases the second NN is trained using one hundred (100) training/testing patterns. For the six storey frame of Figure 5, eight modes are required to capture the 90% of the total mass. For each test case a different neural network configuration is used: (i) NN1: 1-10-8, NN2: 16-20-1, (ii) NN1: 10-20-8, NN2: 16-20-1 and (iii) NN1: 11-20-8, NN2: 16-20-1.

The influence of the three groups of random variables with respect to the number of simulations is show in Figure 8. It can be seen that 2000 - 5000 simulations are required in order to calculate accurately the probability of exceedance of the design limit-state. The life safety limit-state is considered violated if the maximum interstorey drift exceeds 4.0%.

Once an acceptable trained NN in predicting the maximum drift is obtained, the probability of exceedance for each test case is estimated by means of NN based Monte Carlo Simulation. The results for various numbers of simulations are shown in Table 7 for the three test cases examined. It can be seen that the error of the predicted probability of failure with respect to the "exact" one is rather marginal. On the other hand, the computational cost is drastically decreased, approximately 30 times, for all test cases.

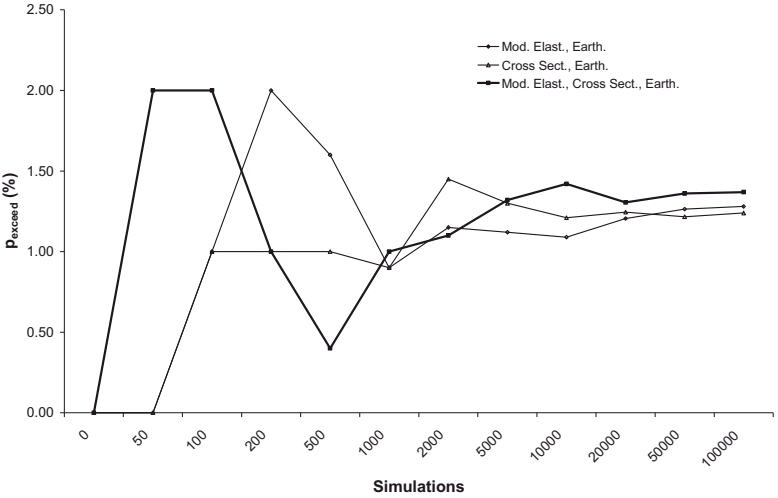


Figure 8. Influence of the number of MC simulations on the value of p_{exceed} for the three test cases

Table 7. "Exact" and predicted values of p_{exceed} and the required CPU time.

	Test case 1		Test case 2		Test case 3	
Number of simulations	"exact" P_{exceed}	NN P_{exceed}	"exact" P_{exceed}	NN P_{exceed}	"exact" P_{exceed}	NN P_{exceed}
50	0.00	0.00	0.00	0.00	2.00	0.00
100	1.00	2.00	1.00	2.00	2.00	1.00
200	2.00	1.60	1.00	1.00	1.00	3.00
500	1.60	1.26	1.00	1.70	0.40	0.70
1,000	0.90	0.81	0.90	0.67	1.00	0.86
2,000	1.15	1.06	1.45	1.41	1.10	0.97
5,000	1.12	1.04	1.30	1.24	1.32	1.18
10,000	1.09	1.03	1.21	1.14	1.42	1.29
20,000	1.21	1.14	1.25	1.31	1.31	1.19
50,000	1.26	1.16	1.22	1.31	1.36	1.21
100,000	1.28	1.16	1.24	1.31	1.37	1.21
CPU time in seconds						
Pattern selection	-	2	-	3	-	5
Training	-	7	-	9	-	12
Propagation	-	25	-	25	-	25
Total	1154	34	1154	37	1154	42

6 Conclusions

An efficient implementation of NN to the fragility analysis of structures based on properly trained neural networks is presented. The neural networks are trained by means of a set of intensity measures that can be easily extracted from

the earthquake records. The methodology of allows the use of Monte Carlo simulation for the calculation of the limit state fragilities, thus avoiding the simplifying assumption that the seismic data follow the lognormal distribution. The presented formulation may be more complicated compared to other simplified approaches, however it offers a different approach to an emerging problem in earthquake engineering leading to reduction of the computational cost. The results obtained once combined with regional hazard curves can be directly applied to the performance-based design of steel frames. On the other hand, the computational effort involved in the conventional Monte Carlo simulation becomes excessive in large-scale problems especially when earthquake loading is considered because of the enormous sample size and the computing time required for each Monte Carlo run. The use of NN can practically eliminate any limitation on the scale of the problem and the sample size used for MCS.

References

Fahlman, S. *An Empirical Study of Learning Speed in Back-Propagation Networks*. Carnegie Mellon: CMU-CS-88-162, 1988.

Hurtado, J.E. Neural network in stochastic mechanics. *Arch. Comp. Meth. Engrg. (State of the art reviews)*, **8**(3): 303-342, 2001.

Hurtado, J.E., Alvarez, D.A. Neural-network-based reliability analysis: a comparative study. *Comp. Meth. Appl. Mech. Engrg.*, **191**: 113-132, 2002.

Kramer, S.L. *Geotechnical Earthquake Engineering*. Prentice-Hall, Englewood Cliffs, NJ, 1996.

Lagaros, N.D., Fragiadakis, M. Fragility assessment of steel frames using neural networks, *Earthquake Spectra*, **23**(4): 735–752, 2007.

Lagaros, N.D., Papadrakakis, M. Learning improvement of neural networks used in structural optimization. *Adv. in Engrg. Software*, **35**: 9-25, 2004.

Lagaros, N.D., Stefanou, G., Papadrakakis, M. An enhanced hybrid method for the simulation of highly skewed non-Gaussian stochastic fields. *Comp. Meth. Appl. Mech. Engrg.*, **194**(45-47): 4824-4844, 2005.

McCorkle, D.S., Bryden, K.M., Carmichael, C.G. A new methodology for evolutionary optimization of energy systems. *Comp. Meth. Appl. Mech. Engrg.*, **192**: 5021-5036, 2003.

Nie, J., Ellingwood, B.R. A new directional simulation method for system reliability. Part II: application of neural networks. *Prob. Engrg. Mech.*., **19**(4): 437-447, 2004.

Papadrakakis, M. Papadopoulos, V., Lagaros, N.D. Structural Reliability analysis of elastic-plastic structures using neural networks and Monte Carlo simulation. *Comp. Meth. Appl. Mech. Engrg.*, **136**: 145-163, 1996.

Papadrakakis, M., Lagaros, N.D. Reliability-based structural optimization using

neural networks and Monte Carlo simulation. *Comp. Methods Appl. Mech. Engrg.*, **191**: 3491-3507, 2002.

Papaioannou, I., Fragiadakis, M., Papadrakakis, M. Inelastic Analysis of Framed Structures using the Fiber Approach. *Proceedings of the 5th International Congress on Computational Mechanics (GRACM 05)*, Limassol, Cyprus, 29 June – 1 July, 2005.

Porter, K.A., Beck, J.L., Shaikhutdinov, R.V. Sensitivity of Building Loss Estimates to Major Uncertain Variables. *Earthquake Spectra*, **18**: 719-743, 2002.

Riedmiller, M. *Advanced Supervised Learning in Multi-layer Perceptrons: From Back-propogation to Adaptive Learning Algorithms.* University of Karlsruhe: W-76128 Karlsruhe, 1994.

Riedmiller, M., Braun, H. A direct adaptive method for faster back-propagation learning: The RPROP algorithm, in H. Ruspini (Ed.), *Proc. of the IEEE International Conference on Neural Networks (ICNN)*, San Francisco, USA, pp. 586-591, 1993.

Schiffmann, W. Joost, M., Werner, R. *Optimization of the back-propagation algorithm for training multi-layer perceptrons.* Technical report, Institute of Physics, University of Koblenz, 1993.

Schuëller, G.I. (ed). Computational Methods in Stochastic Mechanics and Reliability Analysis. *Comp. Meth. Appl. Mech. Engrg. - Special Issue*, **194**(12-16): 1251-1795, 2005.

Shome, N., Cornell, C.A. *Probabilistic seismic demand analysis of non-linear structures*. Report No. RMS-35, RMS Program, Stanford University, Stanford, USA, 1999.

Tsompanakis, Y., Lagaros, N.D., Stavroulakis, G.E. Hybrid soft computing techniques in parameter identification and probabilistic seismic analysis of structures, *Advances in Engineering Software*, **39**: 612-624, 2008.

Vamvatsikos, D., Cornell, C.A. Incremental dynamic analysis. *Eart. Engrg.& Str. Dyn.*, **31**: 491-514, 2002.

Waszczyszyn, Z., Bartczak, M. Neural prediction of buckling loads of cylindrical shells with geometrical imperfections, *International Journal of Non-linear Mechanics*. **37**(4): 763-776, 2002.

Zacharias, J., Hartmann, C., Delgado, A. Damage detection on crates of beverages by artificial neural networks trained with finite-element data. *Comp. Meth. Appl. Mech. Engrg.*, **193**: 561-574, 2004.